Superdeduction at work

Paul Brauner, Clément Houtmann, Claude Kirchner

Superdeduction at Work

Paul Brauner[1], Clément Houtmann[2] and Claude Kirchner[3]

[1] Université Henri Poincaré & LORIA[†]
[2] ENS Cachan & LORIA
[3] INRIA & LORIA

Dedicated to Jean-Pierre Jouannaud on the occasion of his 60th birthday

Abstract Superdeduction is a systematic way to extend a deduction system like the sequent calculus by new deduction rules computed from the user theory. We show how this could be done in a systematic, correct and complete way. We prove in detail the strong normalisation of a proof term language that models appropriately superdeduction. We finaly examplify on several examples, including equality and noetherian induction, the usefulness of this approach which is implemented in the lemuridæ system, written in TOM.

1 Introduction

Our objective is twofold:
– to scale up by an order of magnitude the size of the problems we can deal with;
– to downsize by an order of magnitude the time needed for a given development.
To this end, we started studying a new version of the calculus of constructions in which user-defined computations expressed by rewrite rules can be made transparent in proof terms.
Jean-Pierre Jouannaud [Towards Engineering Proofs, 1999]

The design, verification and communication of formal proofs are central in informatics and mathematics. In the later, the notion of proofs has a long and fruitful history which now becomes even richer with a century of experience in its formalization. In informatics, formal proofs are in particular essential to formaly assess safety as well as security properties of digital systems. In this context, proof engineering becomes crucial and relies on a semi-interactive design where human interaction is unavoidable. Moreover, to be well designed, proofs have to be well understood and built. As the size of critical softwares increases dramatically, typically a "simple" automotive cruise control software consists of more than one hundred thousand lines of code, proof methods and tools should also scale-up.

This proof engineering process is now mastered with the use of proof assistants like Coq[The04], Isabelle [Pau94], PVS [ORS92], HOL [HOL93], Mizar [Rud92] and large libraries of formalised theories ease this task.

[†] UMR 7503 CNRS-INPL-INRIA-Nancy2-UHP. Campus Scientifique, BP 239, 54506 Vandoeuvre-lès-Nancy Cedex, France.

In this context one has to deal with at least two main difficulties. First, proof engineering should scale-up as the theories describing the context become huge and may consist of thousand of axioms and definitions, some of them being quite sophisticated. Second, the proof assistant needs to provide the user with appropriate ways to understand and to guide the proof construction. Both concerns are currently tackled by making libraries available, by providing specific tactics, tacticals or strategies (see typically `coq.inria.fr`), by integration rewriting [BJO02] and decision procedures [NKK02, Alv00, MQP06] safely into the proof assistants, or by interfacing first-order automated theorem provers with proof assistants like [BHdN02] or like the use of Zenon in Focal [Pre05].

Indeed these approaches raise the question of structuring the theories of interest. For instance one would like to identify the subtheory of lists or of naturals to apply specific decision procedures, *e.g.* [KRRT06] and of course finding a good modular structure is one of the first steps in an engineering process.

> *. . . the role of higher-order rewriting is to design a type theoretic frameworks in which computation and deduction are integrated by means of higher-order rewrite rules, while preserving decidability of typing and coherence of the underlying logic . . .*
> *Jean-Pierre Jouannaud [Jou05]*

In this context, we have proposed in [BHK07] a foundational framework making use of three complementary dimensions. First, as pioneered by *deduction modulo*, the computational axioms should be identified. Typically the definition of addition on naturals ought to be embedded into a congruence modulo which deduction is performed [DHK03]. In this case, the deduction rules like the one of natural deduction or of the sequent calculus are not modified but they are applied modulo a congruence embedding part of the theory. Second, we are proposing a complementary approach where *new deduction rules* are inferred from part of the theory in a correct, systematic and complete way. Third, the rest of the theory will be used as the context on which all the standard and new deduction rules will act, possibly modulo some congruence.

To sum up, a theory is split in three parts $\mathcal{T}h = \mathcal{T}h_1 \cup \mathcal{T}h_2 \cup \mathcal{T}h_3$ and instead of seeking for a proof of $\mathcal{T}h_1 \cup \mathcal{T}h_2 \cup \mathcal{T}h_3 \vdash \varphi$, we are building a proof of $\mathcal{T}h_3 \vdash^{+\mathcal{T}h_2}_{\sim_{\mathcal{T}h_1}} \varphi$, *i.e.* we use the theory $\mathcal{T}h_3$ to prove φ using the extended deduction system modulo the congruence $\sim_{\mathcal{T}h_1}$. We assume that the propositions in $\mathcal{T}h_2$ are all proposition rewrite rules, *i.e.* are of the form $\forall \overline{x}.(P \Leftrightarrow \varphi)$, where P is atomic.

To ease the presentation of the main ideas, we will not consider in this paper the case of deduction modulo even if in addition to simplicity it admits unbounded proof size speed-up [Bur07]. We call *superdeduction* the new deduction system embedding the newly generated deduction rules, and the extended entailment relation is denoted $\vdash^{+\mathcal{T}h}$ or simply $\vdash^{+}$.

Intuitively, a superdeduction rule supplants the *folding* of an atomic proposition P by its definition φ, as done by Prawitz [Pra65], followed by as much introductions as possible of the connectives appearing in φ. For instance, the axiom

$$\textsc{Trans} : \forall x.\forall z.(x \leq z \Leftrightarrow \exists y.(x \leq y \wedge y \leq z))$$

is translated into a left deduction rule by first applying the rules of the classical sequent calculus to $\Gamma, \exists y.(x \leq y \wedge y \leq z) \vdash \Delta$. Then by collecting the premises and the side conditions, we get the *new* deduction rule:

$$\leq_{\text{TRANS}\,L} \frac{\Gamma, x \leq y, y \leq z \vdash \Delta}{\Gamma, x \leq z \vdash \Delta}\ y \notin \mathcal{FV}(\Gamma,\Delta)$$

The right rule:

$$\leq_{\text{TRANS}\,R} \frac{\Gamma \vdash x \leq y, \Delta \qquad \Gamma \vdash y \leq z, \Delta}{\Gamma \vdash x \leq z, \Delta}$$

is similarly obtained by applying deduction rules to $\Gamma \vdash \exists y.(x \leq y \wedge y \leq z), \Delta$.

These new deduction rules are quite natural and translate the usual mathematical reasoning *w.r.t.* this axiom. Let us see on a simple example the difference between a proof in sequent calculus and the corresponding one in the extended deduction system. The proof that $\text{TRANS} \vdash a \leq b \Rightarrow b \leq c \Rightarrow a \leq c$ is the following:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\text{Ax}\ \dfrac{}{a \leq b, b \leq c \vdash a \leq b, a \leq c}\quad \text{Ax}\ \dfrac{}{a \leq b, b \leq c \vdash b \leq c, a \leq c}}{\wedge_R\ \dfrac{}{a \leq b, b \leq c \vdash a \leq b \wedge b \leq c, a \leq c}}}{\exists_R\ \dfrac{}{a \leq b, b \leq c \vdash \exists y.(a \leq y \wedge y \leq c), a \leq c}}}{\vdots \qquad \text{Ax}\ \dfrac{}{a \leq c, a \leq b, b \leq c \vdash a \leq c}}}{\Rightarrow_L\ \dfrac{}{\exists y.(a \leq y \wedge y \leq c) \Rightarrow a \leq c, a \leq b, b \leq c \vdash a \leq c}}}{\wedge_L\ \dfrac{}{a \leq c \Leftrightarrow \exists y.(a \leq y \wedge y \leq c), a \leq b, b \leq c \vdash a \leq c}}}{\forall_L\ \dfrac{}{\forall z.(a \leq z \Leftrightarrow \exists y.(a \leq y \wedge y \leq z)), a \leq b, b \leq c \vdash a \leq c}}}{\forall_L\ \dfrac{}{\forall x.\forall z.(x \leq z \Leftrightarrow \exists y.(x \leq y \wedge y \leq z)), a \leq b, b \leq c \vdash a \leq c}}}{\Rightarrow_R\ \dfrac{}{\forall x.\forall z.(x \leq z \Leftrightarrow \exists y.(x \leq y \wedge y \leq z)), a \leq b \vdash b \leq c \Rightarrow a \leq c}}}{\Rightarrow_R\ \dfrac{}{\forall x.\forall z.(x \leq z \Leftrightarrow \exists y.(x \leq y \wedge y \leq z)) \vdash a \leq b \Rightarrow b \leq c \Rightarrow a \leq c}}$$

Using superdeduction, the axiom TRANS has been used to generate the new deduction rules above and the proof becomes simply:

$$\dfrac{\dfrac{\dfrac{\text{Ax}\ \dfrac{}{a \leq b, b \leq c \vdash b \leq c, a \leq c}\quad \text{Ax}\ \dfrac{}{a \leq b, b \leq c \vdash a \leq b, a \leq c}}{\leq_{\text{TRANS}\,R}\ \dfrac{}{a \leq b, b \leq c \vdash a \leq c}}}{\Rightarrow_R\ \dfrac{}{a \leq b \vdash b \leq c \Rightarrow a \leq c}}}{\Rightarrow_R\ \dfrac{}{\vdash a \leq b \Rightarrow b \leq c \Rightarrow a \leq c}}$$

It is important to notice that these new rules are not just "macros" collapsing a sequence of introductions into a single one: they apply to a predicate, not a connector, and therefore do not solely contain purely logical informations. This therefore raises non trivial questions solved in [BHK07] and in this paper, like the conditions under which the system is complete or consistent and sufficient conditions to get cut-elimination.

Superdeduction is based on previous works on supernatural deduction, a deduction system introduced by Benjamin Wack in [Wac05] and providing a logical interpretation of the ρ-calculus [CK01, CLW03]. Preliminary presentation of superdeduction for

the sequent calculus has been given in [Bra06] and the consistency of such systems is studied in [Hou06]. The superdeduction principle has been presented in [BHK07].

In this context, our contributions are the following:

- We first summarize in the next section the general principle defined in [BHK07]: a systematic extension of the classical sequent calculus by new deduction rules inferred from the axioms of the theory that are proposition rewrite rules; We prove in detail that this is correct and complete taking into account permutability problems; Building on Urban's proof-term language for the sequent calculus [Urb01], we present the simple and expressive calculus proposed in [BHK07] that we show to provide a Curry-Howard-de Bruijn correspondence for superdeduction; Assuming the proposition rewrite system used to extend deduction to be weakly normalising and confluent, we prove in detail that the calculus is strongly normalising and therefore that the theory is consistent since the superdeduction system has the cut-elimination property.
- Then, we investigate in Section 3 the consequence of these principles and results for the foundation of a new generation of proof assistants for which we have a first downloadable prototype, lemuridæ (`rho.loria.fr`). In particular we show how convenient and natural proofs become for instance in higher-order logic, mathematical induction, equational logic. We also examplify the current limitations set to get the general results of previous section.
- Finally, we provide in Section 4 the detailed proofs of the results summarized in Section 2.

2 Super sequent calculus

In this section we recall the principles of superdeduction.

$$\textsc{Ax}\ \frac{}{\Gamma, \varphi \vdash \varphi, \Delta} \qquad \textsc{Contr}_R\ \frac{\Gamma \vdash \varphi, \varphi, \Delta}{\Gamma \vdash \varphi, \Delta} \qquad \textsc{Contr}_L\ \frac{\Gamma, \varphi, \varphi \vdash \Delta}{\Gamma, \varphi \vdash \Delta} \qquad \bot_L\ \frac{}{\Gamma, \bot \vdash \Delta}$$

$$\wedge_L\ \frac{\Gamma, \varphi_1, \varphi_2 \vdash \Delta}{\Gamma, \varphi_1 \wedge \varphi_2 \vdash \Delta} \qquad \wedge_R\ \frac{\Gamma \vdash \varphi_1, \Delta \qquad \Gamma \vdash \varphi_2, \Delta}{\Gamma \vdash \varphi_1 \wedge \varphi_2, \Delta} \qquad \top_R\ \frac{}{\Gamma \vdash \top, \Delta}$$

$$\vee_L\ \frac{\Gamma, \varphi_1 \vdash \Delta \qquad \Gamma, \varphi_2 \vdash \Delta}{\Gamma, \varphi_1 \vee \varphi_2 \vdash \Delta} \qquad \vee_R\ \frac{\Gamma \vdash \varphi_1, \varphi_2, \Delta}{\Gamma \vdash \varphi_1 \vee \varphi_2, \Delta} \qquad \Rightarrow_R\ \frac{\Gamma, \varphi_1 \vdash \varphi_2, \Delta}{\Gamma \vdash \varphi_1 \Rightarrow \varphi_2, \Delta}$$

$$\forall_R\ \frac{\Gamma \vdash \varphi, \Delta}{\Gamma \vdash \forall x.\varphi, \Delta}\ x \notin \mathcal{FV}(\Gamma, \Delta) \qquad \forall_L\ \frac{\Gamma, \varphi[t/x] \vdash \Delta}{\Gamma, \forall x.\varphi \vdash \Delta} \qquad \Rightarrow_L\ \frac{\Gamma \vdash \varphi_1, \Delta \qquad \Gamma, \varphi_2 \vdash \Delta}{\Gamma, \varphi_1 \Rightarrow \varphi_2 \vdash \Delta}$$

$$\exists_R\ \frac{\Gamma \vdash \varphi[t/x], \Delta}{\Gamma \vdash \exists x.\varphi, \Delta} \qquad \exists_L\ \frac{\Gamma, \varphi \vdash \Delta}{\Gamma, \exists x.\varphi \vdash \Delta}\ x \notin \mathcal{FV}(\Gamma, \Delta) \qquad \textsc{Cut}\ \frac{\Gamma \vdash \varphi, \Delta \qquad \Gamma, \varphi \vdash \Delta}{\Gamma \vdash \Delta}$$

Figure 1. Classical sequent calculus.

As mentioned in the introduction and similarly as in deduction modulo, we focus our attention to formulæ of the form $\forall \overline{x}.(P \Leftrightarrow \varphi)$ where P is atomic:

Definition 1 (Propositions rewrite rule) *The notation* $\mathsf{R} : P \rightarrow \varphi$ *denotes the axiom* $\forall \overline{x}.(P \Leftrightarrow \varphi)$ *where* R *is a name for it,* P *is an atomic proposition,* φ *some proposition and* $\overline{x}$ *their free variables.*

Notice that P may contain first-order terms and therefore that such an axiom is not just a definition. For instance, $isZero(succ(n)) \rightarrow \bot$ is a proposition rewrite rule.

For the classical sequent calculus, let us now describe how the computation of the superdeduction new inference rules is performed.

Definition 2 (Super sequent calculus rules computation) *Let* $\mathcal{C}alc$ *be a set of rules composed by the subset of the sequent calculus deduction rules formed of* Ax, $\bot_L$, $\top_R$, $\vee_L$, $\vee_R$, $\wedge_L$, $\wedge_R$, $\Rightarrow_L$, $\Rightarrow_R$, $\forall_L$, $\forall_R$, $\exists_L$ *and* $\exists_R$, *as well as of the two following rules* $\top_L$ *and* $\bot_R$

$$\top_L \frac{\Gamma \vdash \Delta}{\Gamma, \top \vdash \Delta} \qquad\qquad \bot_R \frac{\Gamma \vdash \Delta}{\Gamma \vdash \bot, \Delta}$$

Let $\mathsf{R} : P \rightarrow \varphi$ *be a proposition rewrite rule.*

1. *To get the right rule associated with* R*, initialise the procedure with the sequent* $\Gamma \vdash \varphi, \Delta$*. Next, apply the rules of* $\mathcal{C}alc$ *until no more open leave remain on which they can be applied. Then, collect the premises, the side conditions and the conclusion and replace* φ *by* P *to obtain the right rule* R_R*.*
2. *To get the left rule* R_L *associated with* R*, initialise the procedure with the sequent* $\Gamma, \varphi \vdash \Delta$*. apply the rules of* $\mathcal{C}alc$ *and get the new left rule the same way as for the right one.*

Definition 3 (Super sequent calculus) *Given a proposition rewrite system* $\mathcal{R}$*, the super sequent calculus associated with* $\mathcal{R}$ *is formed of the rules of classical sequent calculus and the rules built upon* $\mathcal{R}$*. The sequents in such a system are written* $\Gamma \vdash^{+_{\mathcal{R}}} \Delta$*.*

To ensure good properties of the system, we need to put some restrictions on the axioms though. Although the deduction rules of the classical sequent calculus propositional fragment may be applied in any order to reach axioms, the application order of rules concerning quantifiers is significant. Let us consider the following cases:

$$\forall_R \dfrac{\forall_L \dfrac{\text{Ax}\ \dfrac{}{P(x_0) \vdash P(x_0)}}{\forall x.P(x) \vdash P(x_0)}}{\forall x.P(x) \vdash \forall x.P(x)} \qquad \forall_L \dfrac{P(t) \vdash \forall x.P(x)}{\forall x.P(x) \vdash \forall x.P(x)}$$

The left-hand side proof succeeds because the early application of the $\forall_R$ rule provides the appropriate term for instantiating the variable of the proposition present in the context. On the other hand, the second proof cannot be completed since the $\forall_R$ side condition requires the quantified variable to be substituted for a fresh one. Such a situation may occur when building the super sequent calculus custom rules and therefore

may break its completeness *w.r.t.* classical predicate logic. This common permutability problem of automated proof search appears here since superdeduction systems are in fact embedding a part of compiled automated deduction. Thereby we apply an idea inspired by focusing techniques [And92, AM99, And01], namely replacing every subformula of φ leading to a permutability problem by a fresh predicate symbol parameterised by the free variables of the subformula. To formalise this, we first need to recall the polarity notion:

Definition 4 (Polarity of a subformula) *The polarity $pol_\varphi(\psi)$ of ψ in φ where ψ is a subformula occurrence of φ is a boolean defined as follows:*

- *if $\varphi = \psi$, then $pol_\varphi(\psi) = 1$;*
- *if $\varphi = \varphi_1 \wedge \varphi_2$ or $\varphi_1 \vee \varphi_2$, then $pol_\varphi(\psi) = pol_{\varphi_1}(\psi)$ if ψ is a subformula occurrence of φ_1, $pol_{\varphi_2}(\psi)$ otherwise;*
- *if $\varphi = \forall x.\varphi_1$ or $\exists x.\varphi_1$, then $pol_\varphi(\psi) = pol_{\varphi_1}(\psi)$;*
- *if $\varphi = \varphi_1 \Rightarrow \varphi_2$, then $pol_\varphi(\psi) = \neg pol_{\varphi_1}(\psi)$ if ψ is a subformula occurrence of φ_1, $pol_{\varphi_2}(\psi)$ otherwise.*

Definition 5 (Set of permutability problems) *A formula ψ is in the set $PP(\varphi)$ of φ permutability problems if there exists φ' a subformula of φ such that ψ is a subformula occurrence of φ' and one of these propositions holds:*

- *$\varphi' = \forall x.\varphi'_1$, $\psi = \forall x.\psi'_1$ and $pol_{\varphi'}(\psi) = 0$*
- *$\varphi' = \exists x.\varphi'_1$, $\psi = \exists x.\psi'_1$ and $pol_{\varphi'}(\psi) = 0$*
- *$\varphi' = \forall x.\varphi'_1$, $\psi = \exists x.\psi'_1$ and $pol_{\varphi'}(\psi) = 1$*
- *$\varphi' = \exists x.\varphi'_1$, $\psi = \forall x.\psi'_1$ and $pol_{\varphi'}(\psi) = 1$*

This allows us to define the most appropriate generalisation of a proposition rewrite rule $\mathsf{R} : P \to \varphi$:

Definition 6 (Set of delayed proposition rewrite rules) *This is the set:*

$$Dl(\mathsf{R} : P \to \varphi) = \{P \to C\big[Q_1(\overline{x_1}), \ldots, Q_n(\overline{x_n})\big]\} \bigcup_{i=1\ldots n} Dl\big(Q_i \to \varphi_i\big)$$

such that:

- *C is the largest context in φ with no formula in $PP(\varphi)$ such that $\varphi = C[\varphi_1 \ldots \varphi_n]$;*
- *$\forall i \in \{1 \ldots n\}$, $\overline{x_i}$ is the vector of φ_i free variables;*
- *$Q_1 \ldots Q_n$ are fresh predicate symbols.*

As an example, let us consider the proposition rewrite rule defining the natural numbers as the set of terms verifying the inductive predicate:

$$\in_{\mathbb{N}} : \mathbb{N}(n) \to \forall P.(0 \in P \Rightarrow \forall m.(m \in P \Rightarrow s(m) \in P) \Rightarrow n \in P)$$

This axiom can be found in [DW05] which introduces an axiomatisation of constructive arithmetic with rewrite rules only. It uses a simple second-order encoding by expressing quantification over propositions by quantification over classes; $x \in P$ should therefore

be read as $P(x)$. The delayed set $Dl(\in_{\mathbb{N}})$ of proposition rewrite rules derived from the rules above is:

$$\begin{array}{r} \in_{\mathbb{N}} : \ \mathbb{N}(n) \rightarrow \forall P.(0 \in P \Rightarrow H(P) \Rightarrow n \in P) \\ hered : H(P) \rightarrow \forall m.(m \in P \Rightarrow s(m) \in P) \end{array}$$

Let us notice that the proposition $H(P)$ revealed by the elimination of permutability problems expresses heredity, a well-known notion. Focussing on parts of the propositions which raise some non-trivial choice at some phase on the proof has been naturally done by mathematicians. Then we obtain the following deduction rules for the natural numbers definition:

$$\in_{\mathbb{N}_L} \frac{\Gamma \vdash^+ 0 \in P, \Delta \quad \Gamma \vdash^+ H(P), \Delta \quad \Gamma, n \in P \vdash^+ \Delta}{\Gamma, \mathbb{N}(n) \vdash^+ \Delta}$$

$$\in_{\mathbb{N}_R} \frac{0 \in P, H(P) \vdash^+ n \in P, \Delta}{\Gamma \vdash^+ \mathbb{N}(n), \Delta} \ P \notin \mathcal{FV}(\Gamma, \Delta)$$

The left rule translates exactly the usual induction rule. The $hered$ proposition rewrite rule generates new deduction rules too:

$$hered_L \frac{\Gamma \vdash^+ m \in P, \Delta \qquad \Gamma, s(m) \in P \vdash^+ \Delta}{\Gamma, H(P) \vdash^+ \Delta}$$

$$hered_R \frac{\Gamma, m \in P \vdash^+ s(m) \in P, \Delta}{\Gamma \vdash^+ H(P), \Delta} \ m \notin \mathcal{FV}(\Gamma, \Delta)$$

Once again, the right rule corresponds to the usual semantics of heredity.

Main properties of the super sequent calculus associated with a delayed set of axioms are its soundness and completeness *w.r.t.* classical predicate logic.

Theorem 1 (Soundness and completeness of super sequent calculus) *Given $\mathcal{T}h$ an axiomatic theory made of axioms of the form $\forall \overline{x}.(P \Leftrightarrow \varphi)$ with P atomic and $\mathcal{R}$ the associated proposition rewrite rules, every proof of $\Gamma \vdash_{Dl(\mathcal{R})} \Delta$ in super sequent calculus can be translated into a proof of $\Gamma, \mathcal{T}h \vdash \Delta$ in sequent calculus (soundness) and conversely (completeness).*

Proof. **Soundness.** This is easily proved by replacing every occurrence of a superrule R_R obtained from $P \rightarrow \varphi$ by the partial proof derived during its computation. Then by translating the unfolding step by an application of $\Rightarrow_L$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma, \mathcal{T}h, P \vdash P, \Delta}\,\textsc{Ax} \qquad \dfrac{\pi_R}{\Gamma, \mathcal{T}h, \varphi \vdash \Delta}}{\Gamma, \mathcal{T}h, P \Rightarrow \varphi \vdash P, \Delta}\,\Rightarrow_L}{\Gamma, \mathcal{T}h, P \Leftrightarrow \varphi, P \vdash \Delta}\,\wedge_L}{\cdots}\,\forall_L}{\Gamma, \mathcal{T}h, \forall \overline{x}.(P \Leftrightarrow \varphi), P \vdash \Delta}\,\forall_L}{\Gamma, \mathcal{T}h, P \vdash \Delta}\,\textsc{Contr}_L$$

The left case is symmetric.

Completeness. Let π be the proof of $\Gamma, \mathcal{T}h \vdash \Delta$. By cutting the conclusion on $\mathcal{T}h$, the problem is brought down to proving the axioms of $\mathcal{T}h$ in the super sequent calculus.

$$\textsc{Cut}\ \frac{\dfrac{\ldots}{\Gamma \vdash^{+\mathcal{R}} \mathcal{T}h, \Delta} \qquad \dfrac{\pi}{\Gamma, Th \vdash^{+\mathcal{R}} \Delta}}{\Gamma \vdash^{+\mathcal{R}} \Delta}$$

This is done by induction on the derivations of R_R and R_L for each rewrite rule R of $\mathcal{R}$. The full proof is in [Bra06]. □

A proof-term language for superdeduction has been designed in [BHK07] together with a cut-elimination procedure shown to be strongly normalising under appropriate properties. We will recall its definition now and the full proofs of the strong normalisation property will be written in Section 4. This proof-term language is based upon Christian Urban's work on cut-elimination for classical sequent calculus [Urb00, Urb01, UB01, Len03, vBLL05]. The main difference between Urban's proof-terms and other approaches such as Hugo Herbelin's $\bar{\lambda}\mu\tilde{\mu}$-calculus [Her95, CH00, Wad03] is that no focus is made on a particular formula of a sequent $\Gamma \vdash \Delta$, and thus a proof-term M always annotate the full sequent. Such typing judgements are denoted $M \rhd \Gamma \vdash \Delta$. It is explained in [BHK07] why this difference between Urban's and Herbelin's approaches made us choose the first one to base our proof-terms for superdeduction upon.

Urban's proof-term language for classical sequent calculus makes no use of the first-class objects of the λ-calculus such as *abstractions* or *variables*. Variables are replaced by *names* and *conames*. Let X and A be respectively the set of *names* and the set of *conames*. Symbols $x, y, \ldots$ will range over X while symbols $a, b, \ldots$ will range over A. Symbols $\mathsf{x}, \mathsf{y}, \ldots$ will range over the set of first-order variables. Left-contexts and right-contexts are sets containing respectively pairs $x : \varphi$ and pairs $a : \varphi$. Symbol Γ will range over left-contexts and symbol Δ will range over the right-contexts. Moreover, contexts cannot contain more than one occurrence of a name or coname. We will never omit the 'first-order' in 'first-order term' in order to avoid confusion with 'terms' (*i.e.* proof-terms). The set of terms is defined as follows.

$$\begin{aligned} M, N ::= {} & \mathsf{Ax}(x, a) \mid \mathsf{Cut}(\widehat{a}M, \widehat{x}N) \mid \mathsf{False}_L(x) \mid \mathsf{True}_R(a) \\ \mid {} & \mathsf{And}_R(\widehat{a}M, \widehat{b}N, c) \mid \mathsf{And}_L(\widehat{x}\widehat{y}M, z) \mid \mathsf{Or}_R(\widehat{a}\widehat{b}M, c) \mid \mathsf{Or}_L(\widehat{x}M, \widehat{y}N, z) \\ \mid {} & \mathsf{Imp}_R(\widehat{x}\widehat{a}M, b) \mid \mathsf{Imp}_L(\widehat{x}M, \widehat{a}N, y) \mid \mathsf{Exists}_R(\widehat{a}M, t, b) \mid \mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M, y) \\ \mid {} & \mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}M, b) \mid \mathsf{Forall}_L(\widehat{x}M, t, y) \end{aligned}$$

Names and conames are not called *variables* and *covariables* such as in $\bar{\lambda}\mu\tilde{\mu}$-calculus since they do not represent places where terms might be inserted. They still may appear bound: the symbol «$\widehat{\ }$» is the unique binder of the calculus and thus we can compute the sets of free and bound names, conames and first-order variables in any term. We consequently adopt Barendregt's convention on names, conames and first-order variables: in a term or in a statement a name, a coname or a first-order variable is never both bound and free in the same context.

The type system is expressed in Figure 2. The differences with Urban's type system is the use of $\vee_R \dfrac{\Gamma \vdash \varphi_1, \varphi_2, \Delta}{\Gamma \vdash \varphi_1 \vee \varphi_2, \Delta}$ instead of $\vee_R\text{-}i \dfrac{\Gamma \vdash \varphi_i, \Delta}{\Gamma \vdash \varphi_1 \vee \varphi_2, \Delta}$ for $i \in \{1, 2\}$

and similarly for $\wedge$. A comma in a conclusion stands for the set union and a comma in a premise stands for the *disjoint* set union. This allows our type inference rules to contain *implicit* contraction.

A term M *introduces* the name z if it is of the form $\mathsf{Ax}(z,a)$, $\mathsf{False}_L(z)$, $\mathsf{And}_L(\widehat{x}\widehat{y}M,z)$, $\mathsf{Or}_L(\widehat{x}M,\widehat{y}N,z)$, $\mathsf{Imp}_L(\widehat{x}M,\widehat{a}N,z)$, $\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M,z)$, $\mathsf{Forall}_L(\widehat{x}M,t,z)$, and it *introduces* the coname c is it is of the form $\mathsf{Ax}(x,c)$, $\mathsf{True}_R(c)$, $\mathsf{And}_R(\widehat{a}M,\widehat{b}N,c)$, $\mathsf{Or}_R(\widehat{a}\widehat{b}M,c)$, $\mathsf{Imp}_R(\widehat{x}\widehat{a}M,c)$, $\mathsf{Exists}_R(\widehat{a}M,t,c)$, $\mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}M,c)$. A term M *freshly* introduces a name or a coname if it introduces it, but none of its proper subterms. It means that the corresponding formula is introduced at the top-level of the proof, but not implicitly contracted and consequently introduced in some subproof.

Figure 3 presents a (non-confluent) cut-elimination procedure denoted $\xrightarrow{\mathsf{cut}}$ proven to be strongly normalising on well-typed terms in [Urb00, UB01]. It is *complete* in the sense that irreducible terms are cut-free. $M[b \mapsto a]$ stands for the term M where every free occurrence of the coname b is rewritten to a (and similarly for $Q[y \mapsto x]$). Besides, the proof substitution operation denoted $M[a := \widehat{x}N]$ and its dual $M[x := \widehat{a}N]$ are defined in Figure 4.

$$\textsc{Ax}\ \frac{}{\mathsf{Ax}(x,a) \rhd \Gamma, x:\varphi \vdash a:\varphi,\Delta} \qquad \textsc{Cut}\ \frac{M \rhd \Gamma \vdash a:\varphi,\Delta \qquad N \rhd \Gamma, x:\varphi \vdash \Delta}{\mathsf{Cut}(\widehat{a}M,\widehat{x}N) \rhd \Gamma \vdash \Delta}$$

$$\bot_L\ \frac{}{\mathsf{False}_L(x) \rhd \Gamma, x:\bot \vdash \Delta} \qquad \top_R\ \frac{}{\mathsf{True}_R(a) \rhd \Gamma \vdash a:\top,\Delta}$$

$$\wedge_R\ \frac{M \rhd \Gamma \vdash a:\varphi_1,\Delta \quad N \rhd \Gamma \vdash b:\varphi_2,\Delta}{\mathsf{And}_R(\widehat{a}M,\widehat{b}N,c) \rhd \Gamma \vdash c:\varphi_1\wedge\varphi_2,\Delta} \qquad \wedge_L\ \frac{M \rhd \Gamma, x:\varphi_1, y:\varphi_2 \vdash \Delta}{\mathsf{And}_L(\widehat{x}\widehat{y}M,z) \rhd \Gamma, z:\varphi_1\wedge\varphi_2 \vdash \Delta}$$

$$\vee_R\ \frac{M \rhd \Gamma \vdash a:\varphi_1, b:\varphi_2,\Delta}{\mathsf{Or}_R(\widehat{a}\widehat{b}M,c) \rhd \Gamma \vdash c:\varphi_1\vee\varphi_2,\Delta} \qquad \vee_L\ \frac{M \rhd \Gamma, x:\varphi_1 \vdash \Delta \quad N \rhd \Gamma, y:\varphi_2 \vdash \Delta}{\mathsf{Or}_L(\widehat{x}M,\widehat{y}N,z) \rhd \Gamma, z:\varphi_1\vee\varphi_2 \vdash \Delta}$$

$$\Rightarrow_R\ \frac{M \rhd \Gamma, x:\varphi_1 \vdash a:\varphi_2,\Delta}{\mathsf{Imp}_R(\widehat{x}\widehat{a}M,b) \rhd \Gamma \vdash b:\varphi_1\Rightarrow\varphi_2,\Delta} \qquad \Rightarrow_L\ \frac{M \rhd \Gamma, x:\varphi_2 \vdash \Delta \quad N \rhd \Gamma \vdash a:\varphi_1,\Delta}{\mathsf{Imp}_L(\widehat{x}M,\widehat{a}N,y) \rhd \Gamma, y:\varphi_1\Rightarrow\varphi_2 \vdash \Delta}$$

$$\exists_L\ \frac{M \rhd \Gamma, x:\varphi \vdash \Delta}{\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M,y) \rhd \Gamma, y:\exists \mathsf{x}.\varphi \vdash \Delta}\ \mathsf{x} \notin \mathcal{FV}(\Gamma,\Delta)$$

$$\exists_R\ \frac{M \rhd \Gamma \vdash a:\varphi[\mathsf{x}:=t],\Delta}{\mathsf{Exists}_R(\widehat{a}M,t,b) \rhd \Gamma \vdash b:\exists \mathsf{x}.\varphi,\Delta} \qquad \forall_L\ \frac{M \rhd \Gamma, x:\varphi[\mathsf{x}:=t] \vdash \Delta}{\mathsf{Forall}_L(\widehat{x}M,t,y) \rhd \Gamma, y:\forall \mathsf{x}.\varphi \vdash \Delta}$$

$$\forall_R\ \frac{M \rhd \Gamma \vdash a:\varphi,\Delta}{\mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}M,b) \rhd \Gamma \vdash b:\forall \mathsf{x}.\varphi,\Delta}\ \mathsf{x} \notin \mathcal{FV}(\Gamma,\Delta)$$

Figure 2. Type system.

Now let us extend Urban's proof-term language for superdeduction. During the computation of the deduction rules for some proposition rewrite rule, the procedure

Logical Cuts:

$$\mathsf{Cut}(\widehat{a}M, \widehat{x}\mathsf{Ax}(x,b)) \xrightarrow{\mathsf{cut}} M[a \mapsto b] \qquad \text{if } M \text{ freshly introduces } a$$

$$\mathsf{Cut}(\widehat{a}\mathsf{Ax}(y,a), \widehat{x}M) \xrightarrow{\mathsf{cut}} M[x \mapsto y] \qquad \text{if } M \text{freshly introduces } x$$

$$\mathsf{Cut}(\widehat{a}\mathsf{True}_R(a), \widehat{x}M) \xrightarrow{\mathsf{cut}} M \qquad \text{if } M \text{freshly introduces } x$$

$$\mathsf{Cut}(\widehat{a}M, \widehat{x}\mathsf{False}_L(x)) \xrightarrow{\mathsf{cut}} M \qquad \text{if } M \text{freshly introduces } a$$

$$\mathsf{Cut}(\widehat{a}\mathsf{And}_R(\widehat{b}M_1, \widehat{c}M_2, a), \widehat{x}\mathsf{And}_L(\widehat{y}\widehat{z}N, x)) \xrightarrow{\mathsf{cut}} \begin{cases} \mathsf{Cut}(\widehat{b}M_1, \widehat{y}\mathsf{Cut}(\widehat{c}M_2, \widehat{z}N)) \\ \mathsf{Cut}(\widehat{c}M_2, \widehat{z}\mathsf{Cut}(\widehat{b}M_1, \widehat{y}N)) \end{cases}$$
$$\text{if } \mathsf{And}_R(\widehat{b}M_1, \widehat{c}M_2, a) \text{ and } \mathsf{And}_L(\widehat{y}\widehat{z}N, x) \text{ freshly introduce } a \text{ and } x$$

$$\mathsf{Cut}(\widehat{a}\mathsf{Or}_R(\widehat{b}\widehat{c}M, a), \widehat{x}\mathsf{Or}_L(\widehat{y}N_1, \widehat{z}N_2, x)) \xrightarrow{\mathsf{cut}} \begin{cases} \mathsf{Cut}(\widehat{b}\mathsf{Cut}(\widehat{c}M, \widehat{z}N_2), \widehat{y}N_1) \\ \mathsf{Cut}(\widehat{c}\mathsf{Cut}(\widehat{b}M, \widehat{y}N_1), \widehat{z}N_2) \end{cases}$$
$$\text{if } \mathsf{Or}_R(\widehat{b}\widehat{c}M, a) \text{ and } \mathsf{Or}_L(\widehat{y}N_1, \widehat{z}N_2, x) \text{ freshly introduce } a \text{ and } x$$

$$\mathsf{Cut}(\widehat{a}\mathsf{Imp}_R(\widehat{x}\widehat{b}M, a), \widehat{y}\mathsf{Imp}_L(\widehat{z}N_1, \widehat{c}N_2, y)) \xrightarrow{\mathsf{cut}} \begin{cases} \mathsf{Cut}(\widehat{b}\mathsf{Cut}(\widehat{c}N_2, \widehat{x}M), \widehat{z}N_1) \\ \mathsf{Cut}(\widehat{c}N_2, \widehat{x}\mathsf{Cut}(\widehat{b}M, \widehat{z}N_1)) \end{cases}$$
$$\text{if } \mathsf{Imp}_R(\widehat{x}\widehat{b}M, a) \text{ and } \mathsf{Imp}_L(\widehat{z}N_1, \widehat{c}N_2, y) \text{ freshly introduce } a \text{ and } y$$

$$\mathsf{Cut}(\widehat{a}\mathsf{Exists}_R(\widehat{b}M, t, a), \widehat{x}\mathsf{Exists}_L(\widehat{y}\widehat{\mathsf{x}}N, x)) \xrightarrow{\mathsf{cut}} \mathsf{Cut}(\widehat{b}M, \widehat{y}N[\mathsf{x} := t])$$
$$\text{if } \mathsf{Exists}_R(\widehat{b}M, t, a) \text{ and } \mathsf{Exists}_L(\widehat{y}\widehat{\mathsf{x}}N, x) \text{ freshly introduce } a \text{ and } x$$

$$\mathsf{Cut}(\widehat{a}\mathsf{Forall}_R(\widehat{b}\widehat{\mathsf{x}}M, a), \widehat{x}\mathsf{Forall}_L(\widehat{y}N, t, x)) \xrightarrow{\mathsf{cut}} \mathsf{Cut}(\widehat{b}M[\mathsf{x} := t], \widehat{y}N)$$
$$\text{if } \mathsf{Forall}_R(\widehat{b}\widehat{\mathsf{x}}M, a) \text{ and } \mathsf{Forall}_L(\widehat{y}N, t, x) \text{ freshly introduce } a \text{ and } x$$

Commuting Cuts: $\mathsf{Cut}(\widehat{a}M, \widehat{x}N) \xrightarrow{\mathsf{cut}} \begin{cases} M[a := \widehat{x}N] & \text{if } M \text{ does not freshly introduce } a, \text{ or} \\ N[x := \widehat{a}M] & \text{if } M \text{ does not freshly introduce } x \end{cases}$

Figure 3. Urban's cut-reductions.

computes an *open* derivation where two kinds of information still need to be provided: (1) premises that remain to be proved and (2) first-order terms written at a metalevel by rules $\exists_R$ and $\forall_L$ that still remain to be instantiated. In order to represent these, we use a formal notion of *open-terms*: terms that contains (1) open leaves that represent premises that remain to be proved and are denoted $\Box$, and (2) placeholders for first-order terms that represent uninstantiated first-order terms and are denoted by $\alpha, \beta, \ldots$ Substitutions over placeholder-terms are written $[\alpha := t, \ldots]$ and are defined over first-order terms, formulæ, sequents, and terms. The syntax of open-terms is then:

$$\begin{aligned} C, D ::= {} & \Box \rhd \Gamma \vdash \Delta \mid \mathsf{Ax}(x,a) \mid \mathsf{Cut}(\widehat{a}C, \widehat{x}D) \\ \mid {} & \ldots \\ \mid {} & \mathsf{Exists}_R(\widehat{a}C, \alpha, b) \mid \mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}C, y) \\ \mid {} & \mathsf{Forall}_R(\widehat{a}\widehat{\mathsf{x}}C, b) \mid \mathsf{Forall}_L(\widehat{x}C, \alpha, y) \end{aligned}$$

Urban's cut-elimination procedure is extended to open-terms in the obvious way. Typing is also extended to open-terms by adding the following rule to the type inference rules

$$
\begin{aligned}
\mathsf{Ax}(x,c)[c := \widehat{y}M] &\triangleq M[y \mapsto x]\\
\mathsf{Ax}(y,a)[y := \widehat{c}M] &\triangleq M[c \mapsto a]\\
\mathsf{And}_R(\widehat{a}M_1, \widehat{b}M_2, c)[c := \widehat{y}N] &\triangleq \mathsf{Cut}(\widehat{c}\mathsf{And}_R(\widehat{a}M_1[c := \widehat{y}N], \widehat{b}M_2[c := \widehat{y}N], c), \widehat{y}N)\\
\mathsf{And}_L(\widehat{x}\widehat{y}M, z)[z := \widehat{a}N] &\triangleq \mathsf{Cut}(\widehat{a}N, \widehat{z}\mathsf{And}_L(\widehat{x}\widehat{y}M[z := \widehat{a}N], z))\\
&\dots\\
\mathsf{Exists}_R(\widehat{a}M, t, b)[b := \widehat{x}N] &\triangleq \mathsf{Cut}(\widehat{b}\mathsf{Exists}_R(\widehat{a}M[b := \widehat{x}N], t, b), \widehat{x}N)\\
\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M, y)[y := \widehat{a}N] &\triangleq \mathsf{Cut}(\widehat{a}N, \widehat{y}\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M[y := \widehat{a}N], y))\\
&\dots\\
\text{Otherwise :}&\\
\mathsf{Ax}(x,a)[\vartheta] &\triangleq \mathsf{Ax}(x,a)\\
\mathsf{Cut}(\widehat{a}M, \widehat{x}N)[\vartheta] &\triangleq \mathsf{Cut}(\widehat{a}M[\vartheta], \widehat{x}N[\vartheta])\\
\mathsf{And}_R(\widehat{a}M_1, \widehat{b}M_2, c)[\vartheta] &\triangleq \mathsf{And}_R(\widehat{a}M_1[\vartheta], \widehat{b}M_2[\vartheta], c)\\
\mathsf{And}_L(\widehat{x}\widehat{y}M, z)[\vartheta] &\triangleq \mathsf{And}_L(\widehat{x}\widehat{y}M[\vartheta], z)\\
&\dots\\
\mathsf{Exists}_R(\widehat{a}M, t, b)[\vartheta] &\triangleq \mathsf{Exists}_R(\widehat{a}M[\vartheta], t, b)\\
\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M, y)[\vartheta] &\triangleq \mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}M[\vartheta], y)\\
&\dots
\end{aligned}
$$

Figure 4. Proof Substitution.

of Figure 2.

$$\frac{}{(\Box \rhd \Gamma \vdash \Delta) \rhd \Gamma \vdash \Delta}$$

These leaves will be denoted for short $\dfrac{}{\Box \rhd \Gamma \vdash \Delta}$. Type inference derivation for open-terms are called open type inference derivations. Their *open leaves* are the later leaves, *i.e.* the *open leaves* of the open-term. For some open-term C, its number of occurrences of $\Box$ is denoted n_C. Then for some placeholder-term substitution $\sigma = [\alpha_1 := t_1, \dots, \alpha_p := t_p]$ where all placeholder-terms appearing in C are substituted by σ (we say that σ *covers* C) and for $M_1, \dots, M_{n_C}$ some terms, we define the term $\sigma C[M_1, \dots, M_{n_C}]$ as follows.

- if C is a term and $n_C = 0$ then trivially $\sigma C[] \triangleq \sigma C$;
- if $C = \Box \rhd \Gamma \vdash \Delta$ and $n_C = 1$ then $\sigma C[M] \triangleq M$;
- if $C = \mathsf{And}_R(\widehat{a}C_1, \widehat{b}C_2, c)[M_1, \dots, M_{n_C}]$ then

$$\sigma C[M_1, \dots, M_{n_C}] \triangleq \mathsf{And}_R(\widehat{a}\sigma C_1[M_1, \dots, M_{n_{C_1}}], \widehat{b}\sigma C_2[M_{n_{C_1}+1}, \dots, M_{n_C}], c)\,;$$

- if $C = \mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}C_1, y)$, then

$$\sigma C[M_1, \dots, M_{n_C}] \triangleq \mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}\sigma C_1[M_1, \dots, M_{n_C}], y)\,;$$

- if $C = \mathsf{Exists}_R(\widehat{a}C_1, \alpha, b)$, then

$$\sigma C[M_1, \dots, M_{n_C}] \triangleq \mathsf{Exists}_R(\widehat{a}\sigma C_1[M_1, \dots, M_{n_C}], \sigma\alpha, b)\,;$$

– the other remaining cases are similar.

Let us define now the extended terms and reduction rules associated with the proposition rewrite rule $\mathsf{R} : P \to \varphi$. For some formula φ, for x and a some name and coname, the open-terms denoted $\langle\!|\vdash a : \varphi \,|\!\rangle$ and $\langle\!| \, x : \varphi \vdash |\!\rangle$ are defined as follows.

$$
\begin{aligned}
\langle\!| \, \Gamma \vdash \Delta \, |\!\rangle &\triangleq \Box \rhd \Gamma \vdash \Delta \quad \text{if } \Gamma \text{ and } \Delta \text{ only contain atomic formulæ} \\
\langle\!| \, \Gamma, x : \varphi \vdash a : \varphi, \Delta \, |\!\rangle &\triangleq \mathsf{Ax}(x, a) \\
\langle\!| \, \Gamma \vdash a : \varphi_1 \Rightarrow \varphi_2, \Delta \, |\!\rangle &\triangleq \mathsf{Imp}_R(\widehat{x}\widehat{b}\langle\!| \, \Gamma, x : \varphi_1 \vdash b : \varphi_2, \Delta \, |\!\rangle, a) \\
\langle\!| \, \Gamma, x : \varphi_1 \Rightarrow \varphi_2 \vdash \Delta \, |\!\rangle &\triangleq \mathsf{Imp}_L(\widehat{y}\langle\!| \, \Gamma, y : \varphi_2 \vdash \Delta \, |\!\rangle, \widehat{a}\langle\!| \, \Gamma \vdash a : \varphi_1, \Delta \, |\!\rangle, x) \\
\langle\!| \, \Gamma \vdash a : \varphi_1 \vee \varphi_2, \Delta \, |\!\rangle &\triangleq \mathsf{Or}_R(\widehat{b}\widehat{c}\langle\!| \, \Gamma \vdash b : \varphi_1, c : \varphi_2, \Delta \, |\!\rangle, a) \\
\langle\!| \, \Gamma, x : \varphi_1 \vee \varphi_2 \vdash \Delta \, |\!\rangle &\triangleq \mathsf{Or}_L(\widehat{y}\langle\!| \, \Gamma, y : \varphi_1 \vdash \Delta \, |\!\rangle, \widehat{z}\langle\!| \, \Gamma, z : \varphi_2 \vdash \Delta \, |\!\rangle, x) \\
\langle\!| \, \Gamma \vdash a : \varphi_1 \wedge \varphi_2, \Delta \, |\!\rangle &\triangleq \mathsf{And}_R(\widehat{b}\langle\!| \, \Gamma \vdash b : \varphi_1, \Delta \, |\!\rangle, \widehat{c}\langle\!| \, \Gamma \vdash c : \varphi_2, \Delta \, |\!\rangle, a) \\
\langle\!| \, \Gamma, x : \varphi_1 \vee \varphi_2 \vdash \Delta \, |\!\rangle &\triangleq \mathsf{And}_L(\widehat{y}\widehat{z}\langle\!| \, \Gamma, y : \varphi_1, z : \varphi_2 \vdash \Delta \, |\!\rangle, x) \\
\langle\!| \, \Gamma \vdash a : \exists \mathsf{x}.\varphi, \Delta \, |\!\rangle &\triangleq \mathsf{Exists}_R(\widehat{b}\langle\!| \, \Gamma \vdash b : \varphi[\mathsf{x} := \alpha], \Delta \, |\!\rangle, \alpha, a) \quad \alpha \text{ is fresh} \\
\langle\!| \, \Gamma, x : \exists \mathsf{x}.\varphi \vdash \Delta \, |\!\rangle &\triangleq \mathsf{Exists}_L(\widehat{y}\widehat{\mathsf{x}}\langle\!| \, \Gamma, y : \varphi \vdash \Delta \, |\!\rangle, x) \quad \text{if } \mathsf{x} \notin \mathcal{FV}(\Gamma, \Delta) \\
\langle\!| \, \Gamma \vdash a : \forall \mathsf{x}.\varphi, \Delta \, |\!\rangle &\triangleq \mathsf{Forall}_R(\widehat{b}\widehat{\mathsf{x}}\langle\!| \, \Gamma \vdash b : \varphi, \Delta \, |\!\rangle, a) \quad \text{if } \mathsf{x} \notin \mathcal{FV}(\Gamma, \Delta) \\
\langle\!| \, \Gamma, x : \forall \mathsf{x}.\varphi \vdash \Delta \, |\!\rangle &\triangleq \mathsf{Forall}_L(\widehat{y}\langle\!| \, \Gamma, y : \varphi[\mathsf{x} := \alpha] \vdash \Delta \, |\!\rangle, \alpha, x) \quad \alpha \text{ is fresh}
\end{aligned}
$$

The definition is non-deterministic just as the definition of new deduction rules in super sequent calculus systems. We may pick any of the possibilities just as we do for the computation of new deduction rules.

We prove the following lemma, which states the adequacy of the typing of $\langle\!|\vdash a : \varphi \,|\!\rangle$ (resp. $\langle\!| \, x : \varphi \vdash |\!\rangle$) with the right (resp. left) superdeduction rule associated with a proposition rewrite rule $P \to \varphi$.

Lemma 1 *Let $\mathsf{R} : P \to \varphi$ be some proposition rewrite rule and let C be the open-term $\langle\!|\vdash a : \varphi \,|\!\rangle$. Then, for any instance of the right rule R_R having $\Gamma \vdash a : P, \Delta$ as its conclusion, $C \rhd \Gamma \vdash a : \varphi, \Delta$ is well-typed, and moreover there exists some substitution σ for placeholder-terms covering C such that the sequents in the premises of C substituted by σ are the premises of this instance of R_R.*

Proof. By construction, an instance of R_R can be transformed into a decomposition of the logical connectors of φ, and thus into some open type inference of $C \rhd \Gamma \vdash a : \varphi, \Delta$, by construction of C. The substitution σ substitutes for the placeholder-terms in this open type inference derivation the terms that are used in this instance of R_R. We obtain thus that the sequents in the premises of C substituted by σ are the premises of this instance of R_R. □

An analogous version of Lemma 1 can be proven for the introduction of P on the left. We propose the type inference rules presented as follows for introducing P on the left and on the right.

$$
\mathsf{R}_R \, \frac{\left(\; M_i \rhd \Gamma, x_1^i : A_1^i, \ldots, x_{p_i}^i : A_{p_i}^i \vdash a_1^i : B_1^i, \ldots, a_{q_i}^i : B_{q_i}^i, \Delta \; \right)_{1 \leqslant i \leqslant n}}{\mathsf{R}_R\left(\widehat{\mathsf{x}_1} \ldots \widehat{\mathsf{x}_p}, \left(\widehat{x_1^i} \ldots \widehat{x_{p_i}^i} \, \widehat{a_1^i} \ldots \widehat{a_{q_i}^i} M_i \right)_{1 \leqslant i \leqslant n}, \alpha_1, \ldots, \alpha_q, a \right) \rhd \Gamma \vdash a : P, \Delta} \, \mathcal{C}
$$

n is the number of open leaves of $\langle\!|\vdash a : \varphi \,|\!\rangle$. The side condition $\mathcal{C}$ is the side condition of the corresponding rule in the super sequent calculus. The first-order variables $\mathsf{x}_1, \ldots, \mathsf{x}_p$ are the variables concerned by this side condition and by Lemma 1, they are the bound first-order variables of $\langle\!|\vdash a : \varphi \,|\!\rangle$. The $\alpha_1, \ldots, \alpha_q$ are the placeholder-terms appearing in this later open-term. When using this type inference rule, these placeholder-terms are to be instantiated by first-order terms in the proof-terms as in the formulæ.

$$\mathsf{R}_L \frac{\Big(N_j \rhd \Gamma, y_1^j : C_1^j, \ldots, y_{r_j}^j : C_{r_j}^j \vdash b_1^j : D_1^j, \ldots, b_{s_j}^j : D_{s_j}^j, \Delta \Big)_{1 \leqslant j \leqslant m}}{\mathsf{R}_L\Big(\widehat{\mathsf{y}_1} \ldots \widehat{\mathsf{y}_r}, \Big(\widehat{y_1^j} \ldots \widehat{y_{r_j}^j}\, \widehat{b_1^j} \ldots \widehat{b_{s_j}^j} N_j\Big)_{1 \leqslant j \leqslant m}, \beta_1, \ldots, \beta_s, x\Big) \rhd \Gamma, x : P \vdash \Delta} \mathcal{C}'$$

m is the number of open leaves of $\langle\!|\, x : \varphi \vdash |\!\rangle$. The side condition $\mathcal{C}'$ is the side condition of the corresponding rule in the super sequent calculus. The first-order variables $\mathsf{y}_1, \ldots, \mathsf{y}_r$ are the variables concerned by this side condition and by the version of Lemma 1 for introducing P on the left, they are the bound first-order variables of $\langle\!|\, x : \varphi \vdash |\!\rangle$. The $\beta_1, \ldots, \beta_s$ are the placeholder-terms appearing in this later open-term. By duality it is expected that $p = s$ and $q = r$. When using this type inference rule, these placeholder-terms are to be instantiated by first-order terms in the proof-terms as in the formulæ.

We obtain the extended proof-terms for a super sequent calculus system. Proofs substitutions are extended in the obvious way on proof-terms.

The extended cut-elimination associated with $\xrightarrow{\mathsf{cut}}$, denoted $\xrightarrow{\mathsf{excut}}$, is defined as follows. For each proposition rewrite rule $\mathsf{R} \,:\, P \to \varphi$, for each reduction

$$\mathsf{Cut}(\widehat{a}\langle\!|\vdash a : \varphi \,|\!\rangle, \widehat{x}\langle\!|\, x : \varphi \vdash |\!\rangle) \xrightarrow{\mathsf{cut}}{}^{+} C$$

where C is a normal form for $\xrightarrow{\mathsf{cut}}$, we add to $\xrightarrow{\mathsf{cut}}$ the following rewrite rule.

$$\sigma\mathsf{Cut}\Big(\widehat{a}\mathsf{R}_R\Big(\widehat{\mathsf{x}_1} \ldots \widehat{\mathsf{x}_p}, \Big(\widehat{x_1^i} \ldots \widehat{x_{p_i}^i}\, \widehat{a_1^i} \ldots \widehat{a_{q_i}^i} M_i\Big)_{1 \leqslant i \leqslant n}, \alpha_1 \ldots \alpha_q, a\Big),$$
$$\widehat{x}\mathsf{R}_L\Big(\widehat{\mathsf{y}_1} \ldots \widehat{\mathsf{y}_r}, \Big(\widehat{y_1^j} \ldots \widehat{y_{r_j}^j}\, \widehat{b_1^j} \ldots \widehat{b_{s_j}^j} N_j\Big)_{1 \leqslant j \leqslant m}, \beta_1 \ldots \beta_s, x\Big)\Big)$$
$$\xrightarrow{\mathsf{excut}} \sigma C[M_1, \ldots, N_m]$$

if $\mathsf{R}_R(\ldots)$ and $\mathsf{R}_L(\ldots)$ freshly introduce a and x

Here σ substitutes for each placeholder-term a first-order term. However these terms are *meta* just as the symbol t in the eighth and ninth rules of Figure 2.

The cut-elimination $\xrightarrow{\mathsf{excut}}$ is *complete*: any instance of a cut is a redex and thus a normal form for $\xrightarrow{\mathsf{excut}}$ is cut-free.

An important result of [BHK07] is the following theorem.

Theorem 2 (Strong Normalisation) *Let us suppose that the set of proposition rewrite rules $\mathcal{R}$ is such that for each of its rules $\mathsf{R} : P \to \varphi$:*

- *P contains only first-order variables (no function symbol or constant);*
- *$\mathcal{FV}(\varphi) \subseteq \mathcal{FV}(P)$;*

and such that the rewrite relation $\xrightarrow{\mathsf{prop}}$ associated with $\mathcal{R}$ is weakly normalising and confluent. Then $\xrightarrow{\mathsf{excut}}$ is strongly normalising on well-typed extended terms.

The proof of this theorem is detailed in Section 4. It uses the normal forms of formulæ through the rewrite relation $\xrightarrow{\mathsf{prop}}$ to translate proofs in superdeduction into proofs in usual sequent calculus and thus requires that $\xrightarrow{\mathsf{prop}}$ is weak normalising and confluent. Besides the translation of existential/universal rules requires the two other hypothesis, as it will be explained by a precise counter-example in Section 3.

It is interesting to notice that since Hypothesis 1 implies the cut-admissibility in the super sequent calculus system, and since this system is sound and complete *w.r.t.* predicate logic, it implies the consistency of the corresponding first-order theory.

3 A foundation for new proof assistants

The first strong argument in favour of proof assistants based on superdeduction is the representation of proofs. Indeed, existing proof assistants such as Coq, Isabelle or PVS are based on the proof planning paradigm, where proofs are represented by a succession of applications of tactics and of tacticals. Coq also builds a proof-term, in particular to bring the proof check down to a micro kernel. In these approaches, the witness of the proof is bound to convince the user that the proof is correct but not to actually *explain* it, as usual mathematical proofs often also do. Even if the proof-terms of Coq are displayed as trees or under the form of natural language text, the main steps of the proof are drown in a multitude of usually not expressed logical arguments due to both the underlying calculus and the presence of purely computational parts, *e.g.* the proof that $2 + 3$ equals 5.

Deduction modulo is a first step forward addressing this later issue by internalising computational aspects of a theory inside a congruence. With the canonical rewrite system on naturals, $P(2 + 3) \vdash P(5)$ becomes an axiom. However a congruence defined by proposition rewrite rules whose right-hand side is not atomic does not bring the expected comfort to interactive proving: the choice of a proposition representative in the congruence introduces some nondeterminism which is neither useful nor wanted. Superdeduction solves this problem by narrowing the choice of a deduction rule to the presence in the goal of one of the extended deduction rules conclusions and goes a step further by also eliminating trivial logical arguments in a proof. Thereby, superdeduction provides a framework for naturally building but also communicating and understanding the essence of proofs.

Notice that extended deduction rules contain only atomic premises and conclusions, thus proof building in this system is like plugging in theorems, definitions and axioms together. This points out the fact that logical arguments of proofs are actually encoded by the structure of theorems, which explains why they are usually not mentionned.

Another important aspect of superdeduction is its potential ability to naturally encode custom reasoning schemes. Let us see how superdeduction behaves in practice when confronted to common situations of theorem proving.

3.1 Higher-order logic

An interesting case is the encoding of other logics like higher-order logic which has been expressed through proposition rewrite rules in [Dow97]. As an example, the proposition rewrite rule $\epsilon(\alpha(\dot{\forall}, x)) \rightarrow \forall y.\epsilon(\alpha(x, y))$ is translated into the following deduction rules which mimic the deduction rules of higher-order logic.

$$\frac{\Gamma \vdash^{+} \epsilon(\alpha(x, y)), \Delta}{\Gamma \vdash^{+} \epsilon(\alpha(\dot{\forall}, x)), \Delta}\ (y \notin \mathcal{FV}(\Gamma)) \qquad \frac{\Gamma, \epsilon(\alpha(x, t)) \vdash^{+} \Delta}{\Gamma, \epsilon(\alpha(\dot{\forall}, x)) \vdash^{+} \Delta}$$

The interesting point is that this behaviour is not encoded inside the underlying logic but is the result of the chosen theory which is only a parameter of the system.

3.2 Induction

Another application field of superdeduction is the handling of induction schemes, introduced in Section 2 with the example of structural induction over Peano naturals. Let us carry on this by proving that every natural number is either odd or even in the super sequent calculus. We start by defining the predicates *even* and *odd* with the following three proposition rewrite rules.

$$\begin{aligned} zero &: \quad Even(0) \rightarrow \top \\ even &: Even(s(n)) \rightarrow Odd(n) \\ odd &: \quad Odd(s(n)) \rightarrow Even(n) \end{aligned}$$

This leads to six simple *folding* and *unfolding* rules.

$$zero_L\ \frac{\Gamma \vdash^{+\mathcal{R}} \Delta}{\Gamma, Even(0) \vdash^{+\mathcal{R}} \Delta} \qquad zero_R\ \frac{}{\Gamma \vdash^{+\mathcal{R}} Even(0), \Delta}$$

$$even_L\ \frac{\Gamma, Odd(n) \vdash^{+\mathcal{R}} \Delta}{\Gamma, Even(s(n)) \vdash^{+\mathcal{R}} \Delta} \qquad even_R\ \frac{\Gamma \vdash^{+\mathcal{R}} Odd(n), \Delta}{\Gamma \vdash^{+\mathcal{R}} Even(s(n)), \Delta}$$

$$odd_L\ \frac{\Gamma, Even(n) \vdash^{+\mathcal{R}} \Delta}{\Gamma, Odd(s(n)) \vdash^{+\mathcal{R}} \Delta} \qquad odd_R\ \frac{\Gamma \vdash^{+\mathcal{R}} Even(n), \Delta}{\Gamma \vdash^{+\mathcal{R}} Odd(s(n)), \Delta}$$

Finally, let us recall that the derived inference rules for induction encode second-order reasoning by the use of *classes*, *i.e.* constants standing for propositions. For instance, assuming that the $\dot{odd}$ class represents the Odd predicate, we add the following axiom to the context of the proof : $\forall x.(x \in \dot{odd} \Leftrightarrow Odd(x))$. Here, since we want to prove that every natural is either odd or even, we introduce the $\dot{oe}$ class which encodes the latter proposition. This is done through a proposition rewrite rule:

$$oddoreven\ :\ n \in \dot{oe}\ \rightarrow Odd(n) \vee\ Even(n)$$

This leads to the creation of two new deduction rules for the super sequent calculus.

$$oddoreven_L \; \dfrac{\Gamma, Odd(n) \vdash^{+\mathcal{R}} \Delta \qquad \Gamma, Even(n) \vdash^{+\mathcal{R}} \Delta}{\Gamma, n \in o\dot{o}e \vdash^{+\mathcal{R}} \Delta}$$

$$oddoreven_R \; \dfrac{\Gamma \vdash^{+\mathcal{R}} Odd(n), Even(n), \Delta}{\Gamma \vdash^{+\mathcal{R}} n \in o\dot{o}e, \Delta}$$

We finally can build a proof of $n \in \mathbb{N} \vdash^{+\mathcal{R}} Odd(n) \vee Even(n)$, which is depicted by Figure 5 (some weakening steps are left implicit to lighten the proof tree). Let us call respectively Π_1, Π_2 and Π_3 the premises of the $\in_{\mathbb{N}_L}$ rule. The proof appears to be rather readable compared to a proof of the same proposition in classical sequent calculus: we start by proving that zero is even or odd (Π_1), then that the *even or odd* property is hereditary (Π_2) by using the deduction rules translating the definitions of even, odd and zero. Then we prove that the proposition holds for every integer by using the induction principle expressed by rule $\in_{\mathbb{N}_L}$. The subproof Π_3 is purely axiomatic and would be typically automated in a proof assistant.

$$oddoreven_L \; \dfrac{\text{Ax}\;\dfrac{}{Odd(n) \vdash^{+\mathcal{R}} Odd(n), Even(n)} \qquad \text{Ax}\;\dfrac{}{Even(n) \vdash^{+\mathcal{R}} Odd(n), Even(n)}}{n \in o\dot{o}e \vdash^{+\mathcal{R}} Odd(n), Even(n)}$$

$$hered_R \; \dfrac{oddoreven_R \; \dfrac{oddoreven_L \; \dfrac{even_R \; \dfrac{\text{Ax}\;\dfrac{}{Odd(m) \vdash^{+\mathcal{R}} Odd(m)}}{Odd(m) \vdash^{+\mathcal{R}} Odd(s(m)), Even(s(m))} \qquad odd_R \; \dfrac{\text{Ax}\;\dfrac{}{Even(m) \vdash^{+\mathcal{R}} Even(m)}}{Even(m) \vdash^{+\mathcal{R}} Odd(s(m)), Even(s(m))}}{m \in o\dot{o}e \vdash^{+\mathcal{R}} Odd(s(m)), Even(s(m))}}{m \in o\dot{o}e \vdash^{+\mathcal{R}} s(m) \in o\dot{o}e}}{\vdash^{+\mathcal{R}} H(o\dot{o}e), Odd(n), Even(n)}$$

$$\vee_R \; \dfrac{\in_{\mathbb{N}_L} \; \dfrac{oddoreven_R \; \dfrac{zero_R \; \dfrac{}{\vdash^{+\mathcal{R}} Odd(0), Even(0)}}{\vdash^{+\mathcal{R}} 0 \in o\dot{o}e, Odd(n), Even(n)} \qquad \vdots \qquad \vdots}{n \in \mathbb{N} \vdash^{+\mathcal{R}} Odd(n), Even(n)}}{n \in \mathbb{N} \vdash^{+\mathcal{R}} Odd(n) \vee Even(n)}$$

Figure 5. Proof of $n \in \mathbb{N} \vdash^{+\mathcal{R}} Odd(n) \vee Even(n)$

Let us remark that in a framework mixing superdeduction and deduction modulo, Π_3 would be immediately closed by an axiom, while the encoding of second order by classes could hardly disappear everywhere in the proof tree. Indeed, the proposition

$m \in o\dot{o}e$ for instance would be equal to $Odd(m) \vee Even(m) \vdash^{+\mathcal{R}} Odd(s(m)) \vee Even(s(m))$ modulo $\mathcal{R}$, which would hide the explicit decoding by the successive applications of $oddoreven_R$ and $oddoreven_L$. The study of such a deduction system is an active research topic.

One may argue that this approach is not viable within the framework of proof assistants because it requires to virtually provide a class for each constructible proposition of the language. This would lead to the introduction of an infinite number of constants symbols, as well as an infinity of associated "decoding" axioms. This problem is addressed in [Kir06] which proposes a finite axiomatisation of the theory of classes. The basic idea is to introduce a constant symbol along with its decoding axiom for each predicate symbol of the discourse. They shall be a finite number of them. As an example, let us encode Odd and $Even$:

$$\begin{array}{c} decodeeven : x \in e\dot{v}en \rightarrow Even(x) \\ decodeodd : \ \ x \in \dot{odd} \rightarrow Odd(x) \end{array}$$

However this time, classes encoding complex propositions are *built* over this finite set of constants using function symbols encoding logical connectors. For instance, the $\cup$ function symbol encodes the $\vee$ connector:

$$decodeunion \ : \ x \in a \cup b \rightarrow x \in a \vee x \in b$$

This entails the encoding of the proposition $Odd(x) \vee Even(x)$ by the $x \in \dot{odd} \cup e\dot{v}en$ one. The difficulty of such an approach is the handling of bound variables and predicates arities. This is achieved via the use of De Bruijn indices and axioms distributing variables *a la* explicit substitutions. The latter proposition is eventually encoded by the following term, whose derivation using the decoding axioms is provided here as an example (see [Kir06] for more details):

$$\begin{array}{ll} & x :: nil \in \dot{odd}(1) \cup e\dot{v}en(1) \\ \rightarrow & x :: nil \in \dot{odd}(1) \vee x :: nil \in e\dot{v}en(1) \\ \rightarrow & Odd(1[x :: nil]) \vee x :: nil \in e\dot{v}en(1) \\ \rightarrow & Odd(x) \vee x :: nil \in e\dot{v}en(1) \\ \rightarrow & Odd(x) \vee Even(1[x :: nil]) \\ \rightarrow & Odd(x) \vee Even(x) \end{array}$$

This powerful mechanism enables the simulation of higher-order behaviour in proof assistants in a natural way. Indeed, decoding is only calculus, which therefore is well handled by both deduction modulo and superdeduction. Once again, a system mixing the two approaches would totally hide the encoding part to the user through deduction modulo while providing a natural way of expressing the induction reasoning via an extended deduction rule.

3.3 Equality

Let us see now how superdeduction handles equality. Taken back to the previously discussed higher-order encoding, the Leibniz definition of equality is expressed as follows:

$$eq \ : \ x = y \rightarrow \forall p.(x :: nil \in p \Rightarrow y :: nil \in p)$$

This leads to the derivation of the following new inference rules:

$$eq_L \ \frac{\Gamma \vdash^{+\mathcal{R}} x :: nil \in p, \Delta \qquad \Gamma, y :: nil \in p \vdash^{+\mathcal{R}} \Delta}{\Gamma, x = y \vdash^{+\mathcal{R}} \Delta}$$

$$eq_R \ \frac{\Gamma, x :: nil \in p \vdash^{+\mathcal{R}} y :: nil \in p, \Delta}{\Gamma \vdash^{+\mathcal{R}} x = y, \Delta} \ p \notin \mathcal{FV}(\Gamma, \Delta)$$

The right rule is rather intuitive and is used to prove the reflexivity of equality in two proof steps:

$$\forall_R \ \dfrac{eq_L \ \dfrac{\textsc{Ax} \ \dfrac{}{x :: nil \in p \vdash^{+\mathcal{R}} x :: nil \in p}}{\vdash^{+\mathcal{R}} x = x}}{\vdash^{+\mathcal{R}} \forall x. x = x}$$

The left rule requires a class term encoding a proposition and is typically used to prove extensionality of function symbols. For instance, given a function symbol f, let us prove that $x = y \Rightarrow f(x) = f(y)$ for any x and y. The appropriate proposition to feed the axiom of Leibniz with would then be $f(x) = f(\alpha)$, parameterized by α. Let us translate this into a class term and prove the proposition:

$$eq_L \ \dfrac{\dfrac{\begin{array}{c} eq_R \ \dfrac{\textsc{Ax} \ \dfrac{}{f(x) :: nil \in p \vdash^{+\mathcal{R}} f(x) :: nil \in p}}{\vdash^{+\mathcal{R}} f(x) = f(x)} \\ \vdots \end{array}}{\vdash^{+\mathcal{R}} x :: nil \in f(S(x)) \doteq 1} \qquad \dfrac{\begin{array}{c} \textsc{Ax} \ \dfrac{}{f(x) = f(y) \vdash^{+\mathcal{R}} f(x) = f(y)} \\ \vdots \end{array}}{y :: nil \in f(S(x)) \doteq 1 \vdash^{+\mathcal{R}} f(x) = f(y)}}{x = y \vdash^{+\mathcal{R}} f(x) = f(y)}$$

The dots stands for decoding steps using axioms of [Kir06]. The S function symbol should be read as “shift” and is part of the explicit substitution mechanism.

Thus, while Leibniz’ definition is adapted to proofs of equality metaproperties, simple notions like extensionality require some deduction steps. A natural use of superdeduction would then be to translate this theorem into an inference rule:

$$f_R \ \frac{\Gamma \vdash^{+\mathcal{R}} x = y, \Delta}{\Gamma \vdash^{+\mathcal{R}} f(x) = f(y), \Delta}$$

However, this goes beyond the scope of superdeduction since the proved proposition is not a proposition rewrite rule (*i.e.* an equivalence). A reasonable extension of superdeduction would be the creation of only-right inference rules to translate axioms of

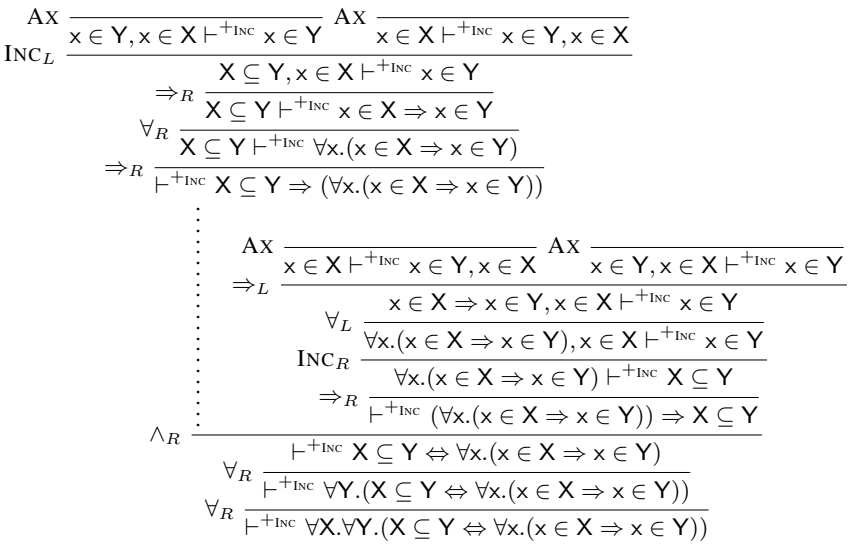

Figure 6. The proof π_1.

the shape $\forall \overline{x}.(P \Rightarrow \varphi)$. Nevertheless, the price to pay would be the loss of the cut-elimination result. The question of extending the cut-elimination procedure to this case is still open.

3.4 Cut-elimination as a translation

An interesting cut-reduction is the following. Let us consider the following proposition rewrite rule:

$$\textsc{Inc} \ : \ \forall A.\forall B.(A \subseteq B \rightarrow \forall \mathsf{x}.(\mathsf{x} \in A \Rightarrow \mathsf{x} \in B))$$

First of all we construct the proof π_1 of $\vdash^{+\textsc{Inc}} \textsc{Inc}$ depicted in Figure 6 (in fact for any theory $\mathcal{T}h$, there is a proof of $\vdash^{+\mathcal{T}h} \mathcal{T}h$ by completeness of superdeduction). The proofterm associated with this proof is

$$\pi_1 = \mathsf{Forall}_R(\widehat{a_2}\widehat{\mathsf{X}}\mathsf{Forall}_R(\widehat{a_3}\widehat{\mathsf{Y}}\mathsf{And}_R(\widehat{a_4}\nu_1, \widehat{a_9}\nu_2, a_3), a_2), a_1)$$

with

$$\nu_1 = \mathsf{Imp}_R(\widehat{x_1}\widehat{a_5}\mathsf{Forall}_R(\widehat{a_6}\widehat{\mathsf{x}}\mathsf{Imp}_R(\widehat{x_2}\widehat{a_7}\textsc{Inc}_L(\widehat{x_3}\mathsf{Ax}(x_3, a_7), \widehat{a_8}\mathsf{Ax}(x_2, a_8), \mathsf{x}, x_1), a_6), a_5), a_4)$$

and

$$\nu_2 = \mathsf{Imp}_R(\widehat{x_4}\widehat{a_{10}}\textsc{Inc}_R(\widehat{\mathsf{x}}\widehat{x_5}\widehat{a_{11}}\mathsf{Forall}_L(\widehat{x_6}\mathsf{Imp}_L(\widehat{x_7}\mathsf{Ax}(x_7, a_{11}), \widehat{a_{12}}\mathsf{Ax}(x_5, a_{12}), x_6), \mathsf{x}, x_4), a_{10}), a_9)$$

Besides we propose the following proof of $\textsc{Inc} \vdash A \subseteq A$, denoted π_2, in raw classical sequent calculus.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\ldots, A \subseteq A \vdash A \subseteq A}\,\textsc{Ax}}{\ldots, (\forall x.(x \in A \Rightarrow x \in A)) \Rightarrow A \subseteq A \vdash A \subseteq A}\,\Rightarrow_L \qquad \dfrac{\dfrac{\dfrac{}{\ldots, x \in A \vdash A \subseteq A, x \in A}\,\textsc{Ax}}{\ldots \vdash A \subseteq A, x \in A \Rightarrow x \in A}\,\Rightarrow_R}{\ldots \vdash A \subseteq A, \forall x.(x \in A \Rightarrow x \in A)}\,\forall_R}{(A \subseteq A) \Leftrightarrow \forall x.(x \in A \Rightarrow x \in A) \vdash A \subseteq A}\,\wedge_L}{\forall Y.(A \subseteq Y) \Leftrightarrow \forall x.(x \in A \Rightarrow x \in Y) \vdash A \subseteq A}\,\forall_L}{\textsc{Inc} \vdash A \subseteq A}\,\forall_L$$

The proofterm associated with this proof is

$$\pi_2 = \mathsf{Forall}_L(\widehat{x_9}\mathsf{Forall}_L(\widehat{x_{10}}\mathsf{And}_L(\widehat{x_{11}}\widehat{x_{12}}\nu_3, x_{10}), A, x_9), A, x_8)$$

with

$$\nu_3 = \mathsf{Imp}_L(\widehat{x_{13}}\mathsf{Ax}(x_{13}, a_{14}), \\ \widehat{a_{15}}\mathsf{Forall}_R(\widehat{a_{16}}\widehat{\mathsf{x}}\mathsf{Imp}_R(\widehat{x_{14}}\widehat{a_{17}}\mathsf{Ax}(x_{14}, a_{17}), a_{16}), a_{15}), x_{12})$$

Now we wish to express the proof π_2 in superdeduction. The corresponding proof denoted π_3 is

$$\dfrac{\dfrac{}{x \in A \vdash^{+\textsc{Inc}} x \in A}\,\textsc{Ax}}{\vdash^{+\textsc{Inc}} A \subseteq A}\,\textsc{Inc}_R$$

We will now obtain it directly from π_2 (and from π_1 whose construction only depends on the axiom INC). Let us consider the proofterm $\mathsf{Cut}(\widehat{a_1}\pi_1, \widehat{x_8}\pi_2)$ which represents the proof

$$\dfrac{\dfrac{\pi_1}{\vdash^{+\textsc{Inc}} \textsc{Inc}} \qquad \dfrac{\pi_2}{\textsc{Inc} \vdash A \subseteq A}}{\vdash^{+\textsc{Inc}} A \subseteq A}\,\textsc{Cut}$$

This proof can also be seen as the *translation* of the proof π_2 in superdeduction: a cut is used to delete the axiom INC from the context. Now it is interesting to understand that the elimination of this cut will actually propagate the superdeduction inference rules contained by π_1 into the proof π_2 and translate the (cut-free) proof of $\textsc{Inc} \vdash A \subseteq A$ into a (cut-free) proof of $\vdash^{+\textsc{Inc}} A \subseteq A$ replacing any use of the axiom INC by a superdeduction rule. An elimination of this cut is depicted in Figure 7. Its result represents the proof π_3.

3.5 Crabbe's counterexample

The (counter)example we consider now is known as *Crabbe's counterexample* and consists in $\mathsf{R} : A \to B \wedge (A \Rightarrow \bot)$. The open-terms associated with it are:

$$\langle\!|\vdash a : B \wedge (A \Rightarrow \bot)\,|\!\rangle = \mathsf{And}_R(\widehat{b}M_1, \widehat{c}\mathsf{Imp}_R(\widehat{x}\widehat{b'}M_2, c), a)$$
$$\langle\!|\, x : B \wedge (A \Rightarrow \bot) \vdash |\!\rangle = \mathsf{And}_L(\widehat{y}\widehat{z}\mathsf{Imp}_L(\widehat{y'}\mathsf{False}_L(y'), \\ \widehat{a}M, z), x)$$

$$
\begin{array}{ll}
 & \mathsf{Cut}(\widehat{a_1}\pi_1, \widehat{x_1}\pi_2) \\
= & \mathsf{Cut}(\widehat{a_1}\mathsf{Forall}_R(\widehat{a_2}\widehat{\mathsf{X}}\mathsf{Forall}_R(\widehat{a_3}\widehat{\mathsf{Y}}\mathsf{And}_R(\widehat{a_4}\nu_1, \widehat{a_9}\nu_2, a_3), a_2), a_1), \\
 & \quad \widehat{x_8}\mathsf{Forall}_L(\widehat{x_9}\mathsf{Forall}_L(\widehat{x_{10}}\mathsf{And}_L(\widehat{x_{11}}\widehat{x_{12}}\nu_3, x_{10}), A, x_9), A, x_8)) \\
\overset{\mathsf{excut}^+}{\longrightarrow} & \mathsf{Cut}(\widehat{a_9}\nu_2, \widehat{x_{12}}\nu_3) \\
= & \mathsf{Cut}(\widehat{a_9}\mathsf{Imp}_R(\widehat{x_4}\widehat{a_{10}}\textsc{Inc}_R(\dots), a_9), \\
 & \quad \widehat{x_{12}}\mathsf{Imp}_L(\widehat{x_{13}}\mathsf{Ax}(x_{13}, a_{14}), \widehat{a_{15}}\mathsf{Forall}_R(\dots), \widehat{x_{12}})) \\
\overset{\mathsf{excut}}{\longrightarrow} & \mathsf{Cut}(\widehat{a_{10}}\mathsf{Cut}(\widehat{a_{15}}\mathsf{Forall}_R(\dots), \widehat{x_4}\textsc{Inc}_R(\dots)), \widehat{x_{13}}\mathsf{Ax}(x_{13}, a_{14})) \\
\overset{\mathsf{excut}}{\longrightarrow} & \mathsf{Cut}(\widehat{a_{15}}\mathsf{Forall}_R(\dots), \widehat{x_4}\textsc{Inc}_R(\widehat{\mathsf{x}\widehat{x_5}\widehat{a_{11}}}\mathsf{Forall}_L(\dots), a_{14})) \\
\overset{\mathsf{excut}}{\longrightarrow} & \textsc{Inc}_R(\widehat{\mathsf{x}\widehat{x_5}\widehat{a_{11}}}\mathsf{Cut}(\widehat{a_{15}}\mathsf{Forall}_R(\dots), \widehat{x_4}\mathsf{Forall}_L(\dots)), a_{14}) \\
\overset{\mathsf{excut}}{\longrightarrow} & \textsc{Inc}_R(\widehat{\mathsf{x}\widehat{x_5}\widehat{a_{11}}}\mathsf{Cut}(\widehat{a_{16}}\mathsf{Imp}_R(\dots), \widehat{x_6}\mathsf{Imp}_L(\dots)), a_{14}) \\
\overset{\mathsf{excut}}{\longrightarrow} & \textsc{Inc}_R(\widehat{\mathsf{x}\widehat{x_5}\widehat{a_{11}}}\mathsf{Cut}(\widehat{a_{12}}\mathsf{Ax}(x_5, a_{12}), \\
 & \quad \widehat{x_{14}}\mathsf{Cut}(\widehat{a_{17}}\mathsf{Ax}(x_{14}, a_{17}), \widehat{x_7}\mathsf{Ax}(x_7, a_{11}))), a_{14}) \\
\overset{\mathsf{excut}^+}{\longrightarrow} & \textsc{Inc}_R(\widehat{\mathsf{x}\widehat{x_5}\widehat{a_{11}}}\mathsf{Ax}(x_5, a_{11}), a_{14})
\end{array}
$$

Figure 7. A cut-elimination of INC

The reduction

$$
\begin{array}{ll}
 & \mathsf{Cut}(\widehat{a}\mathsf{And}_R(\widehat{b}M_1, \widehat{c}\mathsf{Imp}_R(\widehat{x}\widehat{b'}M_2, c), a), \\
 & \quad \widehat{x}\mathsf{And}_L(\widehat{y}\widehat{z}\mathsf{Imp}_L(\widehat{y'}\mathsf{False}_L(y'), \widehat{a}M, z), x)) \\
\overset{\mathsf{cut}}{\longrightarrow}^* & \mathsf{Cut}(\widehat{b}M_1, \widehat{y}\mathsf{Cut}(\widehat{a}M, \widehat{x}M_2))
\end{array}
$$

is replaced by

$$
\begin{array}{ll}
 & \mathsf{Cut}(\widehat{a}\mathsf{R}_R(\widehat{b}M_1, \widehat{x}\widehat{b'}M_2, a), \widehat{x}\mathsf{R}_L(\widehat{y}\widehat{a}M, x)) \\
\longrightarrow & \mathsf{Cut}(\widehat{b}M_1, \widehat{y}\mathsf{Cut}(\widehat{a}M, \widehat{x}M_2))
\end{array}
$$

with *ad hoc* conditions on freshly introduced variables. Let us define the two following terms.

$$
\begin{array}{rl}
\delta & \triangleq \mathsf{R}_L(\widehat{y}\widehat{a}\mathsf{Ax}(x, a), x) \\
\Delta & \triangleq \mathsf{R}_R(\widehat{b}\mathsf{Ax}(z, b), \widehat{x}\widehat{b'}\delta, c)
\end{array}
$$

The following reduction does not terminate:

$$
\begin{array}{ll}
 & \mathsf{Cut}(\widehat{c}\Delta, \widehat{x}\delta) \\
= & \mathsf{Cut}(\widehat{c}\Delta, \widehat{x}\mathsf{R}_L(\widehat{y}\widehat{a}\mathsf{Ax}(x, a), x)) \\
 & \mathsf{R}_L(\widehat{y}\widehat{a}\mathsf{Ax}(x, a), x) \text{ does not freshly introduce } x \\
\longrightarrow & \mathsf{R}_L(\widehat{y}\widehat{a}\mathsf{Ax}(x, a), x)[x := \widehat{c}\Delta] \\
= & \mathsf{Cut}(\widehat{c}\Delta, \widehat{x}\mathsf{R}_L(\widehat{y}\widehat{a}\mathsf{Ax}(x, a)[x := \widehat{c}\Delta], x)) \\
= & \mathsf{Cut}(\widehat{c}\Delta, \widehat{x}\mathsf{R}_L(\widehat{y}\widehat{a}\Delta[c \mapsto a], a)) \\
=_\alpha & \mathsf{Cut}(\widehat{c}\Delta, \widehat{x}\mathsf{R}_L(\widehat{y}\widehat{c}\Delta, a)) \\
= & \mathsf{Cut}(\widehat{c}\mathsf{R}_R(\widehat{b}\mathsf{Ax}(z, b), \widehat{x}\widehat{b'}\delta, c), \widehat{x}\mathsf{R}_L(\widehat{y}\widehat{c}\Delta, a)) \\
\longrightarrow & \mathsf{Cut}(\widehat{c}\mathsf{Cut}(\widehat{b}\mathsf{Ax}(z, b), \widehat{y}\Delta), \widehat{x}\delta) \\
 & \Delta \text{ does not freshly introduces } y \\
\longrightarrow & \mathsf{Cut}(\widehat{c}\Delta[y := \widehat{b}\mathsf{Ax}(z, b)], \widehat{x}\delta) \\
= & \mathsf{Cut}(\widehat{c}\Delta, \widehat{x}\delta) \\
\longrightarrow & \dots
\end{array}
$$

This proposition rewrite rules thus breaks cut-elimination. It obviously does not satisfy Hypothesis 1.

3.6 A convergent presentation of Russel's paradox

This interesting example has first been exposed for deduction modulo in [DW03]. It will be adapted here for superdeduction. Let us consider these two proposition rewrite rules.

$$\mathsf{R}^1 : R \in R \;\rightarrow\; \forall y.(y \simeq R \Rightarrow (R \in y \Rightarrow \bot))$$
$$\mathsf{R}^2 : y \simeq z \;\rightarrow\; \forall y.(x \in y \Rightarrow z \in y)$$

The associated inference rules are

$$\mathsf{R}^1_R\ \frac{\Gamma, \mathsf{y} \simeq R, R \in \mathsf{y} \vdash \Delta}{\Gamma \vdash R \in R, \Delta}\ \mathsf{y} \notin \mathcal{FV}(\Gamma, \Delta) \qquad \mathsf{R}^1_L\ \frac{\Gamma \vdash R \in t, \Delta \qquad \Gamma \vdash t \simeq R, \Delta}{\Gamma, R \in R \vdash \Delta}$$

$$\mathsf{R}^2_R\ \frac{\Gamma, \mathsf{x} \in t_1 \vdash \mathsf{x} \in t_2, \Delta}{\Gamma \vdash t_1 \simeq t_2, \Delta}\ \mathsf{x} \notin \mathcal{FV}(\Gamma, \Delta) \qquad \mathsf{R}^2_L\ \frac{\Gamma, t \in t_2 \vdash \Delta \qquad \Gamma \vdash t \in t_1, \Delta}{\Gamma, t_1 \simeq t_2 \vdash \Delta}$$

Then we can prove $\vdash \bot$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{Ax}}}{R \in R, R \in \mathsf{y} \vdash R \in R, \bot}\mathsf{R}^2_L}{\mathsf{y} \simeq R, R \in \mathsf{y} \vdash R \in R, \bot}}{\vdots}}{\mathsf{y} \simeq R, R \in \mathsf{y}, R \in R \vdash \bot}}{\mathsf{y} \simeq R, R \in \mathsf{y} \vdash \bot}\mathsf{R}^1_R}{\vdash R \in R, \bot}}{\vdots}}{\vdash \bot}\textsc{Cut} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{\mathsf{Ax}}}{R \in R \vdash R \in R, \bot}\mathsf{R}^1_L \qquad \dfrac{\dfrac{\dfrac{}{\mathsf{Ax}}}{R \in R, \mathsf{x} \in R \vdash \mathsf{x} \in R, \bot}\mathsf{R}^2_L}{R \in R \vdash R \simeq R, \bot}}{R \in R \vdash \bot}}{\vdots}}{\vdash \bot}\textsc{Cut}$$

The deduction system is not consistent and since there is no cut-free proof of $\vdash \bot$, strong normalisation of the cut-reduction does not hold. The set of proposition rewrite rules $\{\mathsf{R}^1, \mathsf{R}^2\}$ does not satisfy the hypothesis of Theorem 2 because of the constant R in $R \in R$ which also plays a central role in the proof of $\vdash \bot$.

3.7 lemuridæ

All these properties led us to develop a proof assistant based on the super sequent calculus: lemuridæ. It features extended deduction rules derivation with focussing, rewriting on first-order terms, proof building with the associated superdeduction system, as well as some basic automatic tactics. It is implemented with the TOM [MR06] language, which provides powerful (associative) rewriting capabilities and strategic programmation on top of JAVA. The choice of the TOM language has several beneficial

consequences. First of all, the expressiveness of the language allows for clean and short code. This is in particular the case of the micro proofchecker, whose patterns faithfully translate deduction rules of sequent calculus. Thus, the proofchecker is only one hundred lines long and it is therefore more realistic to convince everyone that it is actually sound.

The other main contribution of TOM to lemuridæ is the expression of tacticals by strategies. The TOM strategy language is directly inspired from early research on ELAN [VB98] and ρ-calculus and allows to compose basic strategies to express complex programs using strategies combinators. In this formalism, a naive proof search tactical is simply expressed by $topdown(elim)$, where $topdown$ is a "call-by-name" strategy and $elim$ has the usual semantics of the corresponding command.

4 Full proofs of the principles

In this section we provide the full proofs of Theorem 2. Let us prove first the following simple result.

Lemma 2 *For some well-typed open-term $C \rhd \Gamma \vdash \Delta$ whose open leaves are $\square \rhd \Gamma_i \vdash \Delta_i$ for $1 \leqslant i \leqslant n_C$, for some σ covering C, if for all $1 \leqslant i \leqslant n_C$, $M_i \rhd \sigma\Gamma_i \vdash \sigma\Delta_i$ is a well-typed term, then $\sigma C[M_1, \ldots, M_{n_C}] \rhd \sigma\Gamma \vdash \sigma\Delta$ is a well-typed term.*

Proof. We proceed by induction on the context C.

- If it is $\square \rhd \Gamma \vdash \Delta$, typed by $\Gamma \vdash \Delta$, then its type inference derivation is the single leaf
$$\frac{}{\square \rhd \Gamma \vdash \Delta}$$
and $n_C = 1$. As by hypothesis $M_1 \rhd \sigma\Gamma \vdash \sigma\Delta$ is well-typed, and as by definition $\sigma C[M_1] = M_1$, $\sigma C[M_1] \rhd \sigma\Gamma \vdash \sigma\Delta$ is well-typed.
- If it is $\mathsf{Ax}(x, a)$, typed by $\Gamma', x : \varphi \vdash a : \varphi, \Delta'$. then its type inference derivation has no leaf since it is
$$\textsc{Ax}\ \frac{}{\mathsf{Ax}(x,a) \rhd \Gamma', x : \varphi \vdash a : \varphi, \Delta'}$$
Then $C = \sigma C[]$ is a term and $\sigma C[] \rhd \sigma\Gamma \vdash \sigma\Delta$ is a well-typed term.
- If it is $\mathsf{And}_R(\widehat{b}C_1, \widehat{c}C_2, a)$, the type inference is
$$\wedge_R\ \frac{\dfrac{\ldots}{C_1 \rhd \Gamma \vdash b : \varphi_1, \Delta'} \qquad \dfrac{\ldots}{C_2 \rhd \Gamma \vdash c : \varphi_2, \Delta'}}{\mathsf{And}_R(\widehat{b}C_1, \widehat{c}C_2, a) \rhd \Gamma \vdash a : \varphi_1 \wedge \varphi_2, \Delta'}$$
By induction hypothesis,
$$\sigma C_1[M_1, \ldots, M_{n_{C_1}}] \rhd \sigma\Gamma, b : \sigma\varphi_1, \sigma\Delta'$$
and
$$\sigma C_2[M_{n_{C_1}+1}, \ldots, M_{n_{C_1}+n_{C_2}}] \rhd \sigma\Gamma, c : \sigma\varphi_2, \sigma\Delta'$$

are well-typed. Then

$$\sigma C[M_1, \ldots, M_{n_C}] \rhd \sigma\Gamma \vdash a : \sigma\varphi_1 \wedge \sigma\varphi_2, \sigma\Delta'$$

is well-typed.

– If it is $\mathsf{Exists}_R(\widehat{a}C_1, \alpha, b)$, the type inference is

$$\exists_R \frac{\dfrac{\ldots}{C_1 \rhd \Gamma \vdash a : \varphi[\mathsf{x} := \alpha], \Delta'}}{\mathsf{Exists}_R(\widehat{a}C_1, \alpha, b) \rhd \Gamma \vdash b : \exists \mathsf{x}.\varphi, \Delta'}$$

By induction hypothesis,

$$\sigma C_1[M_1, \ldots, M_{n_C}] \rhd \sigma\Gamma \vdash a : (\sigma\varphi)[\mathsf{x} := \sigma\alpha], \sigma\Delta'$$

is well-typed and then

$$\sigma C[M_1, \ldots, M_{n_C}] \rhd \sigma\Gamma \vdash b : \exists \mathsf{x}.\sigma\varphi, \sigma\Delta'$$

is well-typed.

– If it is $\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}C_1, y)$, the type inference is

$$\exists_L \frac{\dfrac{\ldots}{C_1 \rhd \Gamma, x : \varphi \vdash \Delta'}}{\mathsf{Exists}_L(\widehat{x}\widehat{\mathsf{x}}C_1, y) \rhd \Gamma, y : \exists \mathsf{x}.\varphi \vdash \Delta'} \; \mathsf{x} \notin \mathcal{FV}(\Gamma, \Delta')$$

By induction hypothesis,

$$\sigma C_1[M_1, \ldots, M_{n_C}] \rhd \sigma\Gamma, x : \sigma\varphi \vdash \sigma\Delta'$$

is well-typed, and then

$$\sigma C[M_1, \ldots, M_{n_C}] \rhd \sigma\Gamma, y : \exists \mathsf{x}.\sigma\varphi \vdash \sigma\Delta'$$

is well-typed.

– other cases are similar.

□

Subject reduction is implied by Lemmas 2 and 1.

Lemma 3 (Subject Reduction) *If $M \overset{\mathsf{excut}^*}{\longrightarrow} M'$ and $M \rhd \Gamma \vdash \Delta$ is well-typed, then $M' \rhd \Gamma \vdash \Delta$ is well-typed.*

Proof. By inspection of the rules defining $\overset{\mathsf{excut}}{\longrightarrow}$. □

We define a rewrite system denoted $\overset{\mathsf{prop}}{\longrightarrow}$ on propositions by turning each proposition rewrite rule into a rewrite rule in the standard way (see for example [DHK03]). We

define a rewrite system denoted $\xrightarrow{\mathsf{term}}$ on extended proof-terms as follows. It contains for each $\mathsf{R} \ : \ P \to \varphi$ the rewrite rule

$$\sigma \mathsf{R}_R\left(\widehat{\mathsf{x}_1} \dots \widehat{\mathsf{x}_p}, \left(\widehat{x_1^i} \dots \widehat{x_{p_i}^i}\, \widehat{a_1^i} \dots \widehat{a_{q_i}^i} M_i\right)_{1 \leqslant i \leqslant n}, \alpha_1 \dots \alpha_q, a\right) \xrightarrow{\mathsf{term}} \sigma \langle\!| \vdash a : \varphi \,|\!\rangle [M_1, \dots, M_n]$$

where σ is a substitution over placeholder-terms covering $\langle\!| \vdash a : \varphi \,|\!\rangle$ (here the bound names and conames of this later open-term are supposed different from the free and bound names and conames of $\mathsf{R}_R(\dots)$) and the rewrite rule

$$\sigma \mathsf{R}_L\left(\widehat{\mathsf{y}_1} \dots \widehat{\mathsf{y}_r}, \left(\widehat{y_1^j} \dots \widehat{y_{r_j}^j}\, \widehat{b_1^j} \dots \widehat{b_{s_j}^j} N_j\right)_{1 \leqslant j \leqslant m}, \beta_1 \dots \beta_s, x\right) \xrightarrow{\mathsf{term}} \sigma \langle\!| \ x : \varphi \vdash |\!\rangle [N_1, \dots, N_m]$$

where σ is a substitution over placeholder-terms covering $\langle\!| \ x : \varphi \vdash |\!\rangle$ (here the bound names and conames of this later open-term are supposed different from the free and bound names and conames of $\mathsf{R}_L(\dots)$).

As $\xrightarrow{\mathsf{term}}$ is orthogonal, it is confluent. Besides if $\xrightarrow{\mathsf{term}}$ is confluent and weakly normalising, then the unique normal form of an extended term M is denoted $M \downarrow^{\mathsf{t}}$. Similarly if $\xrightarrow{\mathsf{prop}}$ is confluent and weakly normalising, then the unique normal form of a formula φ is denoted $\varphi \downarrow^{\mathsf{p}}$. This notation is extended to contexts and sequents. It is also extended to open-terms, since they also contain sequents through the $\Box \rhd \Gamma \vdash \Delta$ constructor.

Let us prove now that $\xrightarrow{\mathsf{excut}}$ is strongly normalising on well-typed extended terms under the following hypothesis.

Hypothesis 1 *For a set of proposition rewrite rules $\mathcal{R}$, the rewrite relation $\xrightarrow{\mathsf{prop}}$ associated with $\mathcal{R}$ is weakly normalising and confluent and for each of its rule $\mathsf{R} : P \to \varphi$:*

- *P contains only first-order variables (no function or constant);*
- *$\mathcal{FV}(\varphi) \subseteq \mathcal{FV}(P)$.*

The second hypothesis restricts the use of first-order constants and functions in particular to avoid counterexamples such as the presentation of Russel's paradox from [DW03] and presented in Section 3 for which the set of proposition rewrite rules terminates but the cut-elimination does not.

Now let us begin our strong normalisation proof with the following lemmas. First, if no proper subterm of M introduces some name or coname and if $M \xrightarrow{\mathsf{term}}{}^{*} M'$, then no proper subterm of M' introduces this name of coname. This remark allows to prove the following lemma.

Lemma 4 *If $M \xrightarrow{\mathsf{term}} M'$ then M freshly introduces some name or coname is equivalent to M' freshly introduces this name of coname.*

By definition of $\xrightarrow{\mathsf{term}}$ with respect to substitutions over first-order variables, the following lemma is straightforward.

Lemma 5 *If $M \xrightarrow{\mathsf{term}} M'$, then for all substitution $[x := t]$, $M[x := t] \xrightarrow{\mathsf{term}} M'[x := t]$. This result extends obviously to $\xrightarrow{\mathsf{term}^*}$.*

This allows to prove the following corollary.

Corollary 1 *If $\xrightarrow{\mathsf{term}}$ is weakly normalising, for all M and $[x := t]$, $(M[x := t])\downarrow^{\mathsf{t}} = (M\downarrow^{\mathsf{t}})[x := t]$.*

Proof. By Lemma 5 and since $M \xrightarrow{\mathsf{term}^*} M\downarrow^{\mathsf{t}}$, then $M[x := t] \xrightarrow{\mathsf{term}^*} (M\downarrow^{\mathsf{t}})[x := t]$. Moreover it is to be noticed that by definition of $\xrightarrow{\mathsf{term}}$ and for all term N, N contains a redex for $\xrightarrow{\mathsf{term}}$ implies that $N[x := t]$ contains a redex. Therefore $(M\downarrow^{\mathsf{t}})[x := t]$ is a normal form for $\xrightarrow{\mathsf{term}}$ and it is $(M[x := t])\downarrow^{\mathsf{t}}$. □

We supposed that in any proposition rewrite rule $\mathsf{R} : P \to \varphi$, P (which is a predicate) only contains first-order variables, and no first-order constant or function. Thus it implies the following lemma.

Lemma 6 *Let φ and φ' be some first-order formulæ such that $\varphi \xrightarrow{\mathsf{prop}} \varphi'$. Let x be some first-order variable and t be some first-order term. Then $\varphi[\mathsf{x} := t] \xrightarrow{\mathsf{prop}} \varphi'[\mathsf{x} := t]$*

Proof. We first suppose that the reduction $\varphi \xrightarrow{\mathsf{prop}} \varphi'$ is done at the head of φ. If the reduction takes place inside a context, we proceed by induction on this context. □

This result is extended to $\xrightarrow{\mathsf{prop}^*}$ in the obvious way. Besides, it implies the following corollary.

Corollary 2 *Let φ be some first-order formula. Let x be some first-order variable and t be some first-order term. Then $(\varphi[\mathsf{x} := t])\downarrow^{\mathsf{p}} = \varphi\downarrow^{\mathsf{p}}[\mathsf{x} := t]$.*

Proof. As $\varphi \xrightarrow{\mathsf{prop}^*} \varphi\downarrow^{\mathsf{p}}$, by Lemma 6, $\varphi[\mathsf{x} := t] \xrightarrow{\mathsf{prop}^*} \varphi\downarrow^{\mathsf{p}}[\mathsf{x} := t]$. If this later formula contains some redex, this redex is an instance of $P(\mathsf{x}_1, \ldots, \mathsf{x}_n)$. Then $\varphi\downarrow^{\mathsf{p}}$ also contains an instance of $P(\mathsf{x}_1, \ldots, \mathsf{x}_n)$. This is a contradiction to the fact that $\varphi\downarrow^{\mathsf{p}}$ is a normal form for $\xrightarrow{\mathsf{prop}}$. Thus $\varphi\downarrow^{\mathsf{p}}[\mathsf{x} := t] = (\varphi[\mathsf{x} := t])\downarrow^{\mathsf{p}}$. □

We can also prove a result similar to Corollary 2 on placeholder-terms substitutions.

Lemma 7 *Let φ be some first-order formula. Let σ be some placeholder-terms substitution. Then $(\sigma\varphi)\downarrow^{\mathsf{p}} = \sigma(\varphi\downarrow^{\mathsf{p}})$.*

Proof. Similar to Corollary 2, with a lemma similar to Lemma 6. □

The last hypothesis we did on the set of proposition rewrite rule is that for each $\mathsf{R} : P \to \varphi$, we have $\mathcal{FV}(\varphi) \subseteq \mathcal{FV}(P)$. It allows to prove the following lemma.

Lemma 8 *Let φ_1 and φ_2 be some formulæ such that $\varphi_1 \xrightarrow{\mathsf{prop}} \varphi_2$. Then $\mathcal{FV}(\varphi_2) \subseteq \mathcal{FV}(\varphi_1)$.*

Proof. – If the reduction $\varphi_1 \xrightarrow{\mathsf{prop}} \varphi_2$ takes place at the head of φ_1. Then for some $\mathsf{R} : P(\mathsf{x}_1, \ldots, \mathsf{x}_p) \to \varphi$, φ_1 is $P(t_1, \ldots, t_p)$ where the t_i are first-order terms. Then φ_2 is $\varphi[(\mathsf{x}_i := t_i)_{1 \leqslant i \leqslant p}]$. As the free variables of φ are by hypothesis included in $\{\mathsf{x}_1, \ldots, \mathsf{x}_p\}$, the free variables of φ_2 are included in $\mathcal{FV}(t_1) \cup \cdots \cup \mathcal{FV}(t_p)$, which is the set $\mathcal{FV}(\varphi_1)$.

– If the reduction $\varphi_1 \xrightarrow{\mathsf{prop}} \varphi_2$ takes place inside a context, we proceed by induction on this context.

□

This result is extended to $\xrightarrow{\mathsf{prop}^*}$ in the obvious way.

Lemma 9 *Any open type inference derivation of $C \rhd \Gamma \vdash \Delta$ with open leaves $\square \rhd \Gamma_i \vdash \Delta_i$ for $1 \leqslant i \leqslant n_C$ may be turned into an open type inference derivation of $C \downarrow^{\mathsf{p}} \rhd \Gamma \downarrow^{\mathsf{p}} \vdash \Delta \downarrow^{\mathsf{p}}$ with premises $\square \rhd \Gamma_i \downarrow^{\mathsf{p}} \vdash \Delta_i \downarrow^{\mathsf{p}}$.*

Proof. By induction on the open type inference derivation.

– One of the base cases is for instance the axiom case : if $C = \mathsf{Ax}(x, a)$, and $C \rhd \Gamma', x : \varphi \vdash a : \varphi, \Delta'$ well-typed (by the axiom rule), then it is straightforward that $C \downarrow^{\mathsf{p}} \rhd \Gamma' \downarrow^{\mathsf{p}}, x : \varphi \downarrow^{\mathsf{p}} \vdash a : \varphi \downarrow^{\mathsf{p}}, \Delta' \downarrow^{\mathsf{p}}$ is well-typed.

– Let us treat the case of an open leaf : if $C = \square \rhd \Gamma \vdash \Delta$, then $C \downarrow^{\mathsf{p}} = \square \rhd \Gamma \downarrow^{\mathsf{p}} \vdash \Delta \downarrow^{\mathsf{p}}$ is also well-typed.

$$\frac{}{(\square \rhd \Gamma \downarrow^{\mathsf{p}} \vdash \Delta \downarrow^{\mathsf{p}}) \rhd \Gamma \downarrow^{\mathsf{p}} \vdash \Delta \downarrow^{\mathsf{p}}}$$

– Let us treat the case of $\wedge_R$. In this case C is $\mathsf{And}_R(\widehat{b}C_1, \widehat{c}C_2, a)$ and the type inference derivation has the following form

$$\wedge_R \dfrac{\dfrac{\dfrac{(\square \rhd \Gamma_i \vdash \Delta_i)_{i \in \{1,\ldots,n_{C_1}\}}}{\cdots}}{C_1 \rhd \Gamma \vdash b : \varphi_1, \Delta'} \qquad \dfrac{\dfrac{(\square \rhd \Gamma_i \vdash \Delta_i)_{i \in \{n_{C_1}+1,\ldots,n_C\}}}{\cdots}}{C_2 \rhd \Gamma \vdash c : \varphi_2, \Delta'}}{C \rhd \Gamma \vdash a : \varphi_1 \wedge \varphi_2, \Delta'}$$

Then by induction hypothesis on the open type inference derivations of C_1 and C_2, we obtain open type inference derivations of $C_1 \downarrow^{\mathsf{p}} \rhd \Gamma \downarrow^{\mathsf{p}} \vdash b : \varphi_1 \downarrow^{\mathsf{p}}, \Delta' \downarrow^{\mathsf{p}}$ and of $C_2 \downarrow^{\mathsf{p}} \rhd \Gamma \downarrow^{\mathsf{p}} \vdash c : \varphi_2 \downarrow^{\mathsf{p}}, \Delta' \downarrow^{\mathsf{p}}$ with open leaves $\square \rhd \Gamma_i \downarrow^{\mathsf{p}} \vdash \Delta_i \downarrow^{\mathsf{p}}$. Finally as $(\varphi_1 \wedge \varphi_2) \downarrow^{\mathsf{p}} = \varphi_1 \downarrow^{\mathsf{p}} \wedge \varphi_2 \downarrow^{\mathsf{p}}$ and $C \downarrow^{\mathsf{p}} = \mathsf{And}_R(\widehat{b}C_1 \downarrow^{\mathsf{p}}, \widehat{c}C_2 \downarrow^{\mathsf{p}}, a)$ this gives using the rule $\wedge_R$ an open type inference derivation of $C \downarrow^{\mathsf{p}} \rhd \Gamma \downarrow^{\mathsf{p}} \vdash \Delta \downarrow^{\mathsf{p}}$.

– Let us treat the case of $\exists_R$. In this case C is $\mathsf{Exists}_R(\widehat{b}C_1, \alpha, a)$ and the type inference derivation has the following form.

$$\exists_R \dfrac{\dfrac{\dfrac{(\square \rhd \Gamma_i \vdash \Delta_i)_{1 \leqslant i \leqslant n_C}}{\cdots}}{C_1 \rhd \Gamma \vdash b : \varphi[\mathsf{x} := \alpha], \Delta'}}{C \rhd \Gamma \vdash a : \exists \mathsf{x}.\varphi, \Delta'}$$

Then by induction hypothesis on the open type inference derivations of C_1, we obtain an open type derivation of $C_1 \downarrow^{\mathsf{p}} \rhd \Gamma \downarrow^{\mathsf{p}} \vdash (\varphi[\mathsf{x} := \alpha]) \downarrow^{\mathsf{p}}, \Delta' \downarrow^{\mathsf{p}}$ with open leaves $\square \rhd \Gamma_i \downarrow^{\mathsf{p}} \vdash \Delta_i \downarrow^{\mathsf{p}}$. By Corollary 2 $(\varphi[\mathsf{x} := \alpha]) \downarrow^{\mathsf{p}}$ is equal to $\varphi \downarrow^{\mathsf{p}} [\mathsf{x} := \alpha]$. Finally as $(\exists \mathsf{x}.\varphi) \downarrow^{\mathsf{p}} = \exists \mathsf{x}.(\varphi) \downarrow^{\mathsf{p}}$ and as $C \downarrow^{\mathsf{p}} = \mathsf{Exists}_R(\widehat{b}C_1, \alpha, a) \downarrow^{\mathsf{p}}$, this give an open type inference derivation of $C \downarrow^{\mathsf{p}} \rhd \Gamma \downarrow^{\mathsf{p}} \vdash \Delta \downarrow^{\mathsf{p}}$.

– Let us treat the case of $\exists_L$. In this case C is $\mathsf{Exists}_L(\widehat{y}\widehat{\mathsf{x}}C_1, x)$ and the type inference derivation has the following form.

$$\exists_L \dfrac{\dfrac{(\square \rhd \Gamma_i \vdash \Delta_i)_{1 \leqslant i \leqslant n_C}}{\dfrac{\cdots}{C_1 \rhd \Gamma', y : \varphi \vdash \Delta}}}{\mathsf{Exists}_L(\widehat{y}\widehat{\mathsf{x}}C_1, x) \rhd \Gamma', x : \exists \mathsf{x}.\varphi \vdash \Delta} \, \mathsf{x} \notin \mathcal{FV}(\Gamma', \Delta)$$

Then by induction hypothesis on the open type inference derivation of C_1, we obtain an open type inference derivation of $C_1 \downarrow^{\mathsf{p}} \rhd \Gamma' \downarrow^{\mathsf{p}}, y : \varphi \downarrow^{\mathsf{p}} \vdash \Delta \downarrow^{\mathsf{p}}$ with open leaves $\square \rhd \Gamma_i \downarrow^{\mathsf{p}} \vdash \Delta_i \downarrow^{\mathsf{p}}$. First of all by Lemma 8 and as $\mathsf{x} \notin \mathcal{FV}(\Gamma', \Delta)$, x is not in $\mathcal{FV}(\Gamma' \downarrow^{\mathsf{p}}, \Delta \downarrow^{\mathsf{p}})$. Furthermore $(\exists \mathsf{x}.\varphi) \downarrow^{\mathsf{p}} = \exists \mathsf{x}.(\varphi) \downarrow^{\mathsf{p}}$. Since $C \downarrow^{\mathsf{p}} = \mathsf{Exists}_L(\widehat{y}\widehat{\mathsf{x}}C_1 \downarrow^{\mathsf{p}}, x)$, we can build an open type inference derivation of $C \downarrow^{\mathsf{p}} \rhd \Gamma \downarrow^{\mathsf{p}} \vdash \Delta \downarrow^{\mathsf{p}}$.

– other cases are similar.

□

Lemma 10 *If $M \rhd \Gamma \vdash \Delta$ is well-typed, then there exists M' such that $M' \rhd \Gamma \downarrow^{\mathsf{p}} \vdash \Delta \downarrow^{\mathsf{p}}$ is well-typed. Besides $M \xrightarrow{\mathsf{term}} M'$ and M' is a normal form, denoted $M \xrightarrow{\mathsf{term!}} M'$.*

Proof. By induction on the type inference derivation of $M \rhd \Gamma \vdash \Delta$.

– If the bottom rule of the derivation is for instance the Ax rule. M is $\mathsf{Ax}(x, a)$ and the derivation is

$$\textsc{Ax} \, \dfrac{}{\mathsf{Ax}(x, a) \rhd \Gamma', x : \varphi \vdash a : \varphi, \Delta'}$$

Then we can build the following derivation.

$$\textsc{Ax} \, \dfrac{}{\mathsf{Ax}(x, a) \rhd \Gamma' \downarrow^{\mathsf{p}}, x : \varphi \downarrow^{\mathsf{p}} \vdash a : \varphi \downarrow^{\mathsf{p}}, \Delta' \downarrow^{\mathsf{p}}}$$

Finally we can check that $M \xrightarrow{\mathsf{term!}} \mathsf{Ax}(x, a)$.

– If the bottom rule of the derivation is for instance the $\wedge_R$ rule. M is $\mathsf{And}_R(\widehat{b}M_1, \widehat{c}M_2, c)$ and the derivation is

$$\wedge_R \, \dfrac{\dfrac{\cdots}{M_1 \rhd \Gamma \vdash b : \varphi_1, \Delta'} \qquad \dfrac{\cdots}{M_2 \rhd \Gamma \vdash c : \varphi_2, \Delta'}}{M \rhd \Gamma \vdash a : \varphi_1 \wedge \varphi_2, \Delta'}$$

By induction hypothesis there exists M_1' and M_2' such that $M_1' \rhd \Gamma \downarrow^{\mathsf{p}} \vdash b : \varphi_1 \downarrow^{\mathsf{p}}, \Delta' \downarrow^{\mathsf{p}}$ and $M_2' \rhd \Gamma \downarrow^{\mathsf{p}} \vdash c : \varphi_2 \downarrow^{\mathsf{p}}, \Delta' \downarrow^{\mathsf{p}}$ are well-typed. Then we can build the

following derivation.

$$\wedge_R \frac{\dfrac{\cdots}{M_1' \rhd \Gamma \downarrow^{\mathsf{p}} \vdash b : \varphi_1 \downarrow^{\mathsf{p}}, \Delta' \downarrow^{\mathsf{p}}} \qquad \dfrac{\cdots}{M_2' \rhd \Gamma \downarrow^{\mathsf{p}} \vdash c : \varphi_2 \downarrow^{\mathsf{p}}, \Delta' \downarrow^{\mathsf{p}}}}{M' \rhd \Gamma \downarrow^{\mathsf{p}} \vdash a : \varphi_1 \downarrow^{\mathsf{p}} \wedge \varphi_2 \downarrow^{\mathsf{p}}, \Delta' \downarrow^{\mathsf{p}}}$$

where M' stands for $\mathsf{And}_R(\widehat{b}M_1', \widehat{c}M_2', c)$. Finally as $\varphi_1 \downarrow^{\mathsf{p}} \wedge \varphi_2 \downarrow^{\mathsf{p}} = (\varphi \wedge \varphi_2) \downarrow^{\mathsf{p}}$ we have found M' such that $M' \rhd \Gamma \downarrow^{\mathsf{p}} \vdash \Delta \downarrow^{\mathsf{p}}$ is well-typed and such that $M \xrightarrow{\mathsf{term!}} M'$.

– If the bottom rule of the derivation is for instance $\exists_R$, M is $\mathsf{Exists}_R(\widehat{b}M_1, t, a)$ and the derivation is

$$\exists_R \frac{\dfrac{\cdots}{M_1 \rhd \Gamma \vdash a : \varphi[\mathsf{x} := t], \Delta'}}{M \rhd \Gamma \vdash a : \exists \mathsf{x}.\varphi, \Delta'}$$

By induction hypothesis there exists M_1' such that $M_1' \rhd \Gamma \downarrow^{\mathsf{p}} \vdash a : \varphi[\mathsf{x} := t] \downarrow^{\mathsf{p}}, \Delta' \downarrow^{\mathsf{p}}$ is well-typed. By Corollary 2, $\varphi[\mathsf{x} := t] \downarrow^{\mathsf{p}} = \varphi \downarrow^{\mathsf{p}} [\mathsf{x} := t]$ and then we can build the derivation.

$$\exists_R \frac{\dfrac{\cdots}{M_1' \rhd \Gamma \downarrow^{\mathsf{p}} \vdash a : \varphi \downarrow^{\mathsf{p}} [\mathsf{x} := t], \Delta' \downarrow^{\mathsf{p}}}}{M' \rhd \Gamma \downarrow^{\mathsf{p}} \vdash a : \exists \mathsf{x}.\varphi \downarrow^{\mathsf{p}}, \Delta' \downarrow^{\mathsf{p}}}$$

where M' stands for $\mathsf{Exists}_R(\widehat{b}M_1', t, a)$. Finally as $(\exists \mathsf{x}.\varphi) \downarrow^{\mathsf{p}} = \exists \mathsf{x}.\varphi \downarrow^{\mathsf{p}}$, we have found M' such that $M' \rhd \Gamma \downarrow^{\mathsf{p}} \vdash \Delta \downarrow^{\mathsf{p}}$ is well-typed and $M \xrightarrow{\mathsf{term!}} M'$.

– If the bottom rule of the derivation is for instance $\exists_L$, M is $\mathsf{Exists}_L(\widehat{y}\widehat{\mathsf{x}}M_1, x)$ and the derivation is

$$\exists_L \frac{\dfrac{\cdots}{M_1 \rhd \Gamma', y : \varphi \vdash \Delta}}{M \rhd \Gamma', x : \exists \mathsf{x}.\varphi \vdash \Delta} \; \mathsf{x} \notin \mathcal{FV}(\Gamma', \Delta)$$

By induction hypothesis there exists M_1' such that $M_1' \rhd \Gamma' \downarrow^{\mathsf{p}}, x : \varphi \downarrow^{\mathsf{p}} \vdash \Delta \downarrow^{\mathsf{p}}$ is well-typed. As $\mathsf{x} \notin \mathcal{FV}(\Gamma', \Delta)$ and by Lemma 8, $\mathsf{x} \notin \mathcal{FV}(\Gamma' \downarrow^{\mathsf{p}}, \Delta \downarrow^{\mathsf{p}})$, we can build the following derivation.

$$\exists_L \frac{\dfrac{\cdots}{M_1' \rhd \Gamma' \downarrow^{\mathsf{p}}, y : \varphi \downarrow^{\mathsf{p}} \vdash \Delta \downarrow^{\mathsf{p}}}}{M' \rhd \Gamma' \downarrow^{\mathsf{p}}, x : \exists \mathsf{x}.\varphi \downarrow^{\mathsf{p}} \vdash \Delta \downarrow^{\mathsf{p}}} \; \mathsf{x} \notin \mathcal{FV}(\Gamma' \downarrow^{\mathsf{p}}, \Delta \downarrow^{\mathsf{p}})$$

where M' stands for $\mathsf{Exists}_L(\widehat{y}\widehat{\mathsf{x}}M_1', x)$. Finally as $\exists \mathsf{x}.\varphi \downarrow^{\mathsf{p}} = \exists \mathsf{x}.\varphi \downarrow^{\mathsf{p}}$, we have found M' such as $M' \rhd \Gamma \downarrow^{\mathsf{p}} \vdash \Delta \downarrow^{\mathsf{p}}$ is well-typed and $M \xrightarrow{\mathsf{term!}} M'$.

– If the bottom rule of the derivation is not an extended rule, other cases are similar.

– If the bottom rule of the derivation is an extended rule, say R_R for $\mathsf{R} : P \to \varphi$, it has the form

$$\mathsf{R}_R \frac{(M_i \rhd \Gamma_i \vdash \Delta_i)_i}{\mathsf{R}_R(\ldots, (\ldots M_i)_i, \ldots, a) \rhd \Gamma \vdash a : P, \Delta'} \; \mathcal{C}$$

Let us denote $C = \langle\!\vdash a : \varphi \,\rangle\!\rangle$. By induction hypothesis there exists $M'_1, \ldots, M'_{n_C}$ such that for all i, $M'_i \rhd \Gamma_i \downarrow^{\mathsf{p}} \vdash \Delta_i \downarrow^{\mathsf{p}}$ is well-typed and $M_i \xrightarrow{\mathsf{term!}} M'_i$. Besides by Lemma 1, there exists a substitution for placeholder-terms σ and an open type inference derivation whose open leaves are the $\square \rhd \Gamma'_i \vdash \Delta'_i$ with $\sigma\Gamma'_i = \Gamma_i$ and $\sigma\Delta'_i = \Delta_i$ for all i and whose conclusion is $C \rhd \Gamma \vdash a : \varphi, \Delta'$. By Lemma 9, this open type inference derivation can be turned into one with open leaves $\square \rhd \Gamma'_i \downarrow^{\mathsf{p}} \vdash \Delta'_i \downarrow^{\mathsf{p}}$ and with conclusion $C \downarrow^{\mathsf{p}} \rhd \Gamma \downarrow^{\mathsf{p}} \vdash a : \varphi \downarrow^{\mathsf{p}}, \Delta' \downarrow^{\mathsf{p}}$. Let us notice that for all i and by Lemma 7, $\Gamma_i \downarrow^{\mathsf{p}} = (\sigma\Gamma'_i) \downarrow^{\mathsf{p}} = \sigma(\Gamma'_i \downarrow^{\mathsf{p}})$ and $\Delta_i \downarrow^{\mathsf{p}} = (\sigma\Delta'_i) \downarrow^{\mathsf{p}} = \sigma(\Delta'_i \downarrow^{\mathsf{p}})$. Thus by Lemma 2, $\sigma C \downarrow^{\mathsf{p}} [(M'_i)_i] \rhd \sigma(\Gamma \downarrow^{\mathsf{p}}) \vdash a : \sigma(\varphi \downarrow^{\mathsf{p}}), \sigma(\Delta' \downarrow^{\mathsf{p}})$ is well-typed. Since $\sigma\Gamma \downarrow^{\mathsf{p}} = \Gamma \downarrow^{\mathsf{p}}$, $\sigma\varphi \downarrow^{\mathsf{p}} = \varphi \downarrow^{\mathsf{p}}$ and $\sigma\Delta' \downarrow^{\mathsf{p}} = \Delta' \downarrow^{\mathsf{p}}$ (Γ, φ and Δ' appear in a derivation in the super sequent calculus and therefore do not contain placeholder-terms !) and since $P \downarrow^{\mathsf{p}} = \varphi \downarrow^{\mathsf{p}}$, this is a type inference of $\sigma C \downarrow^{\mathsf{p}} [(M'_i)_i] \rhd \Gamma \downarrow^{\mathsf{p}} \vdash a : P \downarrow^{\mathsf{p}}, \Delta' \downarrow^{\mathsf{p}}$. Finally as for all i, $M_i \xrightarrow{\mathsf{term!}} M'_i$, then

$$\begin{aligned} M &= \mathsf{R}_R(\ldots, (\ldots M_i)_i, \ldots, a) \\ &\xrightarrow{\mathsf{term}} \sigma C[(M_i)_i] = \sigma C \downarrow^{\mathsf{p}} [(M_i)_i] \\ &\xrightarrow{\mathsf{term}} \sigma C \downarrow^{\mathsf{p}} [(M'_i)_i] \end{aligned}$$

As this later term is a normal form, $M \xrightarrow{\mathsf{term!}} \sigma C \downarrow^{\mathsf{p}} [(M'_i)_i]$.

□

Corollary 3 *$\xrightarrow{\mathsf{term}}$ is weakly normalising on well-typed extended terms. Moreover for all $M \rhd \Gamma \vdash \Delta$ well-typed, $M \downarrow^{\mathsf{t}} \rhd \Gamma \downarrow^{\mathsf{p}} \vdash \Delta \downarrow^{\mathsf{p}}$ is well-typed in Urban's type system.*

Proof. From Lemma 10. □

Lemma 11 *If $M \xrightarrow{\mathsf{excut}} M'$, then $M \downarrow^{\mathsf{t}} \xrightarrow{\mathsf{cut}}{}^{+} M' \downarrow^{\mathsf{t}}$.*

Proof. Let us suppose first that the reduction $M \xrightarrow{\mathsf{excut}} M'$ is done at the head of M. We can distinguish two cases.

– if the reduction is a $\xrightarrow{\mathsf{cut}}$ reduction, then M is a redex for the $\xrightarrow{\mathsf{cut}}$ reduction. Let us consider for instance the $\wedge$ case. Thus M has the form

$$\mathsf{Cut}(\widehat{a}\mathsf{And}_R(\widehat{b}M_1, \widehat{c}M_2, a), \widehat{x}\mathsf{And}_L(\widehat{y}\widehat{z}N, x))$$

where $\mathsf{And}_R(\widehat{b}M_1, \widehat{c}M_2, a)$ and $\mathsf{And}_L(\widehat{y}\widehat{z}N, x)$ freshly introduces a and x and M' may have the form

$$\mathsf{Cut}(\widehat{b}M_1, \widehat{y}\mathsf{Cut}(\widehat{c}M_2, \widehat{z}N)) \text{ (case 1)}$$

or the form

$$\mathsf{Cut}(\widehat{c}M_2, \widehat{z}\mathsf{Cut}(\widehat{b}M_1, \widehat{y}N)) \text{ (case 2)}$$

Then $M \downarrow^{\mathsf{t}}$ is

$$\mathsf{Cut}(\widehat{a}\mathsf{And}_R(\widehat{b}M_1 \downarrow^{\mathsf{t}}, \widehat{c}M_2 \downarrow^{\mathsf{t}}, a), \widehat{x}\mathsf{And}_L(\widehat{y}\widehat{z}N \downarrow^{\mathsf{t}}, x))$$

where $\mathsf{And}_R(\widehat{b}M_1\ {\downarrow^{\mathsf{t}}}, \widehat{c}M_2\ {\downarrow^{\mathsf{t}}}, a)$ and $\mathsf{And}_L(\widehat{y}\widehat{z}N\ {\downarrow^{\mathsf{t}}}, x)$ freshly introduces a and x (Lemma 4) and reduces in one step into

$$\mathsf{Cut}(\widehat{b}M_1\ {\downarrow^{\mathsf{t}}}, \widehat{y}\mathsf{Cut}(\widehat{c}M_2\ {\downarrow^{\mathsf{t}}}, \widehat{z}N\ {\downarrow^{\mathsf{t}}}))$$

and also into

$$\mathsf{Cut}(\widehat{c}M_2\ {\downarrow^{\mathsf{t}}}, \widehat{z}\mathsf{Cut}(\widehat{b}M_1\ {\downarrow^{\mathsf{t}}}, \widehat{y}N\ {\downarrow^{\mathsf{t}}}))$$

The first is $M'\ {\downarrow^{\mathsf{t}}}$ in case 1, the second is $M'\ {\downarrow^{\mathsf{t}}}$ in case 2. So in both cases, $M\ {\downarrow^{\mathsf{t}}} \xrightarrow{\mathsf{cut}}{}^{+}\ M'\ {\downarrow^{\mathsf{t}}}$.

– If the reduction is a $\xrightarrow{\mathsf{cut}}$ reduction, let us consider for instance the $\exists$ case. Thus M has the form

$$\mathsf{Cut}(\widehat{a}\mathsf{Exists}_R(\widehat{b}M, t, a), \widehat{x}\mathsf{Exists}_L(\widehat{y}\widehat{\mathsf{x}}N, x))$$

where $\mathsf{Exists}_R(\widehat{b}M, t, a)$ freshly introduces a and M' is

$$\mathsf{Cut}(\widehat{b}M, \widehat{y}N[\mathsf{x} := t])$$

Then $M\ {\downarrow^{\mathsf{t}}}$ is

$$\mathsf{Cut}(\widehat{a}\mathsf{Exists}_R(\widehat{b}M\ {\downarrow^{\mathsf{t}}}, t, a), \widehat{x}\mathsf{Exists}_L(\widehat{y}\widehat{\mathsf{x}}N\ {\downarrow^{\mathsf{t}}}, x))$$

where $\mathsf{Exists}_R(\widehat{b}M\ {\downarrow^{\mathsf{t}}}, t, a)$ freshly introduces a (Lemma 4) and reduces in one step into

$$\mathsf{Cut}(\widehat{b}M, \widehat{y}N\ {\downarrow^{\mathsf{t}}}\ [\mathsf{x} := t])$$

By Corollary 1, $N\ {\downarrow^{\mathsf{t}}}\ [\mathsf{x} := t] = (N[\mathsf{x} := t])\ {\downarrow^{\mathsf{t}}}$ and we obtain that the later one-step reduct of $M\ {\downarrow^{\mathsf{t}}}$ is in fact $M'\ {\downarrow^{\mathsf{t}}}$.

– If the reduction is a $\xrightarrow{\mathsf{cut}}$ reduction, let us consider the case where M is

$$\mathsf{Cut}(\widehat{a}M_1, \widehat{x}M_2)$$

with M_1 does not freshly introduce a (the case where M_2 does not freshly introduce x is symmetrical) and M' is

$$M_1[a := \widehat{x}M_2]$$

Then $M\ {\downarrow^{\mathsf{t}}}$ is

$$\mathsf{Cut}(\widehat{a}M_1\ {\downarrow^{\mathsf{t}}}, \widehat{x}M_2\ {\downarrow^{\mathsf{t}}})$$

and since $M_1\ {\downarrow^{\mathsf{t}}}$ does not freshly introduce a (Lemma 4), we deduce that it reduces to

$$M_1\ {\downarrow^{\mathsf{t}}}\ [a := \widehat{x}M_2\ {\downarrow^{\mathsf{t}}}]$$

As this later is a normal form and a reduct of M' for $\xrightarrow{\mathsf{term}}$, it is $M'\ {\downarrow^{\mathsf{t}}}$.

– Other cases of $\xrightarrow{\mathsf{cut}}$ reductions are similar.

– If the reduction is a $\xrightarrow{\mathsf{excut}}$ reduction, then M is of the form

$$\mathsf{Cut}(\widehat{a}\mathsf{R}_R(\ldots, (\ldots M_i)_i, \ldots, a), \widehat{x}\mathsf{R}_L(\ldots, (\ldots N_j)_j, \ldots, x))$$

with $\mathsf{R} : P \to \varphi$. Let us denote C_R and C_L respectively $\langle\!|\vdash a : \varphi\ |\!\rangle$ and $\langle\!|\ x : \varphi \vdash |\!\rangle$. Thus we may write the following reduction in $\xrightarrow{\mathsf{term}}$.

$$\begin{aligned} M &= \mathsf{Cut}(\widehat{a}\mathsf{R}_R((\dots M_i)_i, a), \widehat{x}\mathsf{R}_L((\dots N_j)_j, x)) \\ &\xrightarrow{\mathsf{term}} \mathsf{Cut}(\widehat{a}\sigma C_R[(M_i)_i], \widehat{x}\sigma' C_L[(N_j)_j]) \\ &\xrightarrow{\mathsf{term}} \mathsf{Cut}(\widehat{a}\sigma C_R[(M_i\downarrow^{\mathsf{t}})_i], \widehat{x}\sigma' C_L[(N_j\downarrow^{\mathsf{t}})_j]) \end{aligned}$$

where σ and σ' are *ad hoc* placeholder-term substitutions. As this later term is a normal form for $\xrightarrow{\mathsf{term}}$, it is in fact $M\downarrow^{\mathsf{t}}$. Besides by definition of $\xrightarrow{\mathsf{excut}}$, there exists an open-term C such that $\mathsf{Cut}(\widehat{a}C_R, \widehat{x}C_L) \xrightarrow{\mathsf{cut}}^{+} C$ with $M' = \sigma''C[M_1, \dots, N_p]$, and thus $M'\downarrow^{\mathsf{t}} = \sigma''C[M_1\downarrow^{\mathsf{t}}, \dots, N_p\downarrow^{\mathsf{t}}]$. As $\mathsf{Cut}(\widehat{a}C_R, \widehat{x}C_L) \xrightarrow{\mathsf{cut}}^{+} C$, we deduce finally that $M\downarrow^{\mathsf{t}} \xrightarrow{\mathsf{cut}}^{+} M'\downarrow^{\mathsf{t}}$.

Now let us suppose that the reduction $M \xrightarrow{\mathsf{excut}} M'$ is done under some context. We reason by induction on this context. We just treated the case of an empty context.

- Let us consider now for instance the case of R_R. M is of the form $\mathsf{R}_R(\dots, (\dots, M_i)_i, \dots, a)$ and M' is $\mathsf{R}_R(\dots, (\dots, M'_i)_i, \dots, a)$ with some k such that $M_k \xrightarrow{\mathsf{excut}} M'_k$ and for all $i \neq k$, $M'_i = M_i$. By induction hypothesis, $M_k\downarrow^{\mathsf{t}} \xrightarrow{\mathsf{cut}}^{+} M'_k\downarrow^{\mathsf{t}}$ and then
$$\begin{aligned} M\downarrow^{\mathsf{t}} &= \sigma C[(M_i\downarrow^{\mathsf{t}})_i] \\ &\xrightarrow{\mathsf{cut}}^{+} \sigma C[(M'_i\downarrow^{\mathsf{t}})_i] \\ &= M'\downarrow^{\mathsf{t}} \end{aligned}$$
- Let us consider now for instance the case of And_R. M is of the form $\mathsf{And}_R(\widehat{b}M_1, \widehat{c}M_2, a)$ and M' is of the form $\mathsf{And}_R(\widehat{b}M'_1, \widehat{c}M'_2, a)$ with some i in $\{1, 2\}$ such that $M_i \xrightarrow{\mathsf{excut}} M'_i$ and $M_k = M'_k$ for $k \neq i$. By induction hypothesis, $M_i\downarrow^{\mathsf{t}} \xrightarrow{\mathsf{cut}}^{+} M'_i\downarrow^{\mathsf{t}}$ and thus
$$\begin{aligned} M\downarrow^{\mathsf{t}} &= \mathsf{And}_R(\widehat{b}M_1\downarrow^{\mathsf{t}}, \widehat{c}M_2\downarrow^{\mathsf{t}}, a) \\ &\xrightarrow{\mathsf{cut}}^{+} \mathsf{And}_R(\widehat{b}M'_1\downarrow^{\mathsf{t}}, \widehat{c}M'_2\downarrow^{\mathsf{t}}, a) \\ &= M'\downarrow^{\mathsf{t}} \end{aligned}$$
- Let us consider now for instance the case Exists_R. M is of the form $\mathsf{Exists}_R(\widehat{b}M_1, t, a)$ and M' is of the form $\mathsf{Exists}_R(\widehat{b}M'_1, t, a)$ with $M_1 \xrightarrow{\mathsf{excut}} M'_1$. By induction hypothesis, $M_1\downarrow^{\mathsf{t}} \xrightarrow{\mathsf{cut}}^{+} M'_1\downarrow^{\mathsf{t}}$ and thus
$$\begin{aligned} M\downarrow^{\mathsf{t}} &= \mathsf{Exists}_R(\widehat{b}M_1\downarrow^{\mathsf{t}}, t, a) \\ &\xrightarrow{\mathsf{cut}}^{+} \mathsf{Exists}_R(\widehat{b}M'_1\downarrow^{\mathsf{t}}, t, a) \\ &= M'\downarrow^{\mathsf{t}} \end{aligned}$$
- Let us consider now for instance the case Exists_L. M is of the form $\mathsf{Exists}_L(\widehat{y}\widehat{\mathsf{x}}M_1, x)$ and M' is $\mathsf{Exists}_L(\widehat{y}\widehat{\mathsf{x}}M'_1, x)$ with $M_1 \xrightarrow{\mathsf{excut}} M'_1$. By induction

hypothesis, $M_1 \downarrow^{\mathsf{t}} \xrightarrow{\mathsf{cut}}{}^{+} M_1' \downarrow^{\mathsf{t}}$ and thus

$$\begin{array}{rcl} M \downarrow^{\mathsf{t}} & = & \mathsf{Exists}_L(\widehat{y}\widehat{x}M_1 \downarrow^{\mathsf{t}}, x) \\ & \xrightarrow{\mathsf{cut}} & \mathsf{Exists}_L(\widehat{y}\widehat{x}M_1' \downarrow^{\mathsf{t}}, x) \\ & = & M' \downarrow^{\mathsf{t}} \end{array}$$

– Other cases are similar.

□

Now we can prove the main result:

Theorem 2 (Strong Normalisation) *If the set of proposition rewrite rules satisfies Hypothesis 1, then* $\xrightarrow{\mathsf{excut}}$ *is strongly normalising on well-typed extended terms.*

Proof. Let us suppose that $\xrightarrow{\mathsf{prop}}$ is convergent. Let $M \rhd \Gamma \vdash \Delta$ be some well-typed extended term. Let us suppose that there exists an infinite reduction

$$M = M_0 \xrightarrow{\mathsf{excut}} M_1 \xrightarrow{\mathsf{excut}} M_2 \ldots$$

First by Corollary 3, $\xrightarrow{\mathsf{term}}$ is weakly normalising and $M \downarrow^{\mathsf{t}} \rhd \Gamma \downarrow^{\mathsf{p}} \vdash \Delta \downarrow^{\mathsf{p}}$. Besides by Lemma 11, there is an infinite reduction

$$M \downarrow^{\mathsf{t}} = M_0 \downarrow^{\mathsf{t}} \xrightarrow{\mathsf{cut}}{}^{+} M_1 \downarrow^{\mathsf{t}} \xrightarrow{\mathsf{cut}}{}^{+} M_2 \downarrow^{\mathsf{t}} \ldots$$

This is impossible since $M \downarrow^{\mathsf{t}}$ is well-typed in Urban's calculus and $\xrightarrow{\mathsf{cut}}$ is strongly normalising on well-typed terms [Urb00]. □

5 Conclusion

We have motivated and presented superdeduction, a powerful systematic way of extending deduction systems with rules derived from an axiomatic theory. First, we have presented its application to classical sequent calculus along with its properties. After having exhibited a proof-term language associated with this deduction system along with a cut-elimination procedure, we have shown in details its strong normalisation under non-trivial hypothesis, therefore ensuring the consistency of a large class of theories, as well as of the corresponding instances of the system. We have shown on significative examples including higher-order logic, induction and equality why superdeduction could be a grounding framework for a new generation of interactive proof environments. A prototype of this framework, lemuridæ, has been presented and can be actually downloaded.

The very promising results obtained when using lemuridæ, first in term of proof discovery agility and second in the close relationship between human constructed proofs and superdeduction ones, are all very encouraging and trigger the further development of the concepts and implementation. This leads to new questions, since, as seen in Section 3, the behavior of superdeduction systems with propositions considered modulo a congruence is important to study now in details. This will for instance allow building

proofs modulo the symmetry of equality. Another promising point of further research is program extraction from lemuridæ proof-terms along with a computational interpretation of extended deduction rules. We anticipate the extracted programs to have modular structures inherited from the superdeduction proof.

The link, studied in[BDW07], between supernatural deduction (e.g. superdeduction applied to natural deduction) and natural deduction modulo, shows the equivalence between strong normalisation of cut elimination in supernatural deduction and in natural deduction modulo for the implicational fragment of predicate logic. The links between cut elimination in superdeduction and deduction modulo for the sequent calculus have still to be worked out. However, we already can import theories expressed by proposition rewrite rules for deduction modulo to super sequent calculus systems. This is in particular the case of Peano's arithmetic [DW05], but also of Zermelo-Frænkel axiomatization of set theory [DM07].

Finally, let us stress out the recent encoding of pure types systems in $\lambda\Pi$-calculus modulo [CD07]. Indeed, since recent works by G. Burel show that the $\lambda\Pi$-calculus can be naturally encoded in the super sequent calculus, this globally confirms the legitimacy of superdeduction as a foundation for high-level proof assistants. It opens also new questions on the global architecture of proof systems as well as on the interaction with users, either humans or programs.

Acknowledgments: Many thanks to Benjamin Wack for inspiring discussions and his seminal work on this topics, to Horatiu Cirstea for his detailed and crisp comments on previous version this work, to Dan Dougherty for helpful discussions, to the Modulo meetings and the Protheo team for many interactions.

References

[Alv00] C. Alvarado. Reflection for rewriting in the calculus of inductive constructions. In *Proceedings of TYPES 2000*, Durham, United Kingdom, December 2000.

[AM99] J.-M. Andreoli and R. Maieli. Focusing and proof-nets in linear and non-commutative logic. In *LPAR*, volume 1705 of *Lecture Notes in Computer Science*, pages 321–336. Springer, 1999.

[And92] J.-M. Andreoli. Logic programming with focusing proofs in linear logic. *Journal of Logic and Computation*, 2(3):297–347, 1992.

[And01] J.-M. Andreoli. Focussing and proof construction. *Annals Pure Applied Logic*, 107(1-3):131–163, 2001.

[BDW07] P. Brauner, G. Dowek, and B. Wack. Normalization in supernatural deduction and in deduction modulo. Available at `http://hal.inria.fr/inria-00141720`, 2007.

[BHdN02] M. Bezem, D. Hendriks, and H. de Nivelle. Automated proof construction in type theory using resolution. *Journal of Automated Reasoning*, 29(3-4):253–275, 2002.

[BHK07] P. Brauner, C. Houtmann, and C. Kirchner. Principles of superdeduction. In *Proceedings of LICS*, July 2007.

[BJO02] F. Blanqui, J.-P. Jouannaud, and M. Okada. Inductive Data Type Systems. *Theoretical Computer Science*, 272(1-2):41–68, 2002.

[Bra06] P. Brauner. Un calcul des séquents extensible. Master's thesis, Université Henri Poincaré – Nancy 1, 2006.

[Bur07] G. Burel. Unbounded proof-length speed-up in deduction modulo. Technical report, INRIA Lorraine, 2007. Available at http://hal.inria.fr/inria-00138195.

[CD07] D. Cousineau and G. Dowek. Embedding pure type systems in the lambda-pi-calculus modulo. In *TLCA*, 2007. To appear.

[CH00] P.-L. Curien and H. Herbelin. The duality of computation. In *ICFP '00: Proceedings of the fifth ACM SIGPLAN international conference on Functional programming*, pages 233–243, New York, NY, USA, 2000. ACM Press.

[CK01] H. Cirstea and C. Kirchner. The rewriting calculus — Part I *and* II. *Logic Journal of the Interest Group in Pure and Applied Logics*, 9(3):427–498, May 2001.

[CLW03] H. Cirstea, L. Liquori, and B. Wack. Rewriting calculus with fixpoints: Untyped and first-order systems. In *Proceedings of TYPES*, volume 3085 of *Lecture Notes in Computer Science*. Springer, 2003.

[DHK03] G. Dowek, T. Hardin, and C. Kirchner. Theorem proving modulo. *Journal of Automated Reasoning*, 31(1):33–72, Nov 2003.

[DM07] G. Dowek and A. Miquel. Cut elimination for Zermelo's set theory. Available on author's web page, 2007.

[Dow97] G. Dowek. Proof normalization for a first-order formulation of higher-order logic. In E. Gunter and A. Felty, editors, *TPHOL*, volume 1275 of *Lecture Notes in Computer Science*, pages 105–119. Springer, 1997.

[DW03] G. Dowek and B. Werner. Proof normalization modulo. *Journal of Symbolic Logic*, 68(4):1289–1316, 2003.

[DW05] G. Dowek and B. Werner. Arithmetic as a theory modulo. In J. Giesl, editor, *Proceedings of RTA'05*, volume 3467 of *Lecture Notes in Computer Science*, pages 423–437. Springer, 2005.

[Her95] H. Herbelin. *Séquents qu'on calcule*. PhD thesis, Université Paris 7, January 1995.

[HOL93] University of Cambridge, DSTO, SRI Internatioal. *Description of the HOL-System*, 1993.

[Hou06] C. Houtmann. Cohérence de la déduction surnaturelle. Master's thesis, École Normale Supérieure de Cachan, 2006.

[Jou05] J.-P. Jouannaud. Higher-order rewriting: Framework, confluence and termination. In A. Middeldorp, V. van Oostrom, F. van Raamsdonk, and R. C. de Vrijer, editors, *Processes, Terms and Cycles*, pages 224–250, 2005.

[Kir06] F. Kirchner. A finite first-order theory of classes. http://www.lix.polytechnique.fr/Labo/Florent.Kirchner/, 2006.

[KRRT06] H. Kirchner, S. Ranise, C. Ringeissen, and D.-K. Tran. Automatic combinability of rewriting-based satisfiability procedures. In M. Hermann and A. Voronkov, editors, *LPAR*, volume 4246 of *Lecture Notes in Computer Science*, pages 542–556. Springer, 2006.

[Len03] S. Lengrand. Call-by-value, call-by-name, and strong normalization for the classical sequent calculus. *Electronic Notes in Theoretical Computer Science*, 86(4), 2003.

[MQP06] J. Meng, C. Quigley, and L. C. Paulson. Automation for interactive proof: First prototype. *Information and Computation*, 204(10):1575—1596, 2006.

[MR06] P.-E. Moreau and A. Reilles. The tom home page. http://tom.loria.fr, 2006.

[NKK02] Q.-H. Nguyen, C. Kirchner, and H. Kirchner. External rewriting for skeptical proof assistants. *Journal of Automated Reasoning*, 29(3-4):309–336, 2002.

[ORS92] S. Owre, J. M. Rushby, , and N. Shankar. PVS: A prototype verification system. In D. Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga, NY, jun 1992. Springer-Verlag.

[Pau94] L. Paulson. *Isabelle: A Generic Theorem Prover*, volume 828 of *Lecture Notes in Computer Science*. Springer-Verlag, 1994.

[Pra65] D. Prawitz. *Natural Deduction. A Proof-Theoretical Study*, volume 3 of *Stockholm Studies in Philosophy*. Almqvist & Wiksell, Stockholm, 1965.

[Pre05] V. Prevosto. Certified mathematical hierarchies: the focal system. In T. Coquand, H. Lombardi, and M.-F. Roy, editors, *Mathematics, Algorithms, Proofs*, number 05021 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2005. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany.

[Rud92] P. Rudnicki. An overview of the Mizar project. Notes to a talk at the workshop on *Types for Proofs and Programs*, June 1992.

[The04] The Coq development team. *The Coq proof assistant reference manual*. LogiCal Project, 2004. Version 8.0.

[UB01] C. Urban and G. M. Bierman. Strong normalisation of cut-elimination in classical logic. *Fundam. Inform.*, 45(1-2):123–155, 2001.

[Urb00] C. Urban. *Classical Logic and Computation*. PhD thesis, University of Cambridge, October 2000.

[Urb01] C. Urban. Strong normalisation for a gentzen-like cut-elimination procedure. In *TLCA*, pages 415–430, 2001.

[VB98] E. Visser and Z.-e.-A. Benaissa. A core language for rewriting. In C. Kirchner and H. Kirchner, editors, *WRLA*, volume 15 of *Electronic Notes in Theoretical Computer Science*, Pont-à-Mousson, France, sep 1998. Elsevier.

[vBLL05] S. van Bakel, S. Lengrand, and P. Lescanne. The language $\mathcal{X}$: circuits, computations and classical logic. In M. Coppo, E. Lodi, and G. M. Pinna, editors, *Proceedings of Ninth Italian Conference on Theoretical Computer Science (ICTCS'05), Siena, Italy*, volume 3701 of *Lecture Notes in Computer Science*, pages 81–96. Springer, 2005.

[Wac05] B. Wack. *Typage et déduction dans le calcul de réécriture*. PhD thesis, Université Henri Poincaré, Nancy 1, October 2005.

[Wad03] P. Wadler. Call-by-value is dual to call-by-name. In *ICFP '03: Proceedings of the eighth ACM SIGPLAN international conference on Functional programming*, volume 38-9, pages 189–201. ACM Press, September 2003.